Adv. Appl. Probab. **56,** 1307–1338 (2024)
*doi:*10.1017/apr.2023.62

PERFECT SAMPLING OF STOCHASTIC MATCHING MODELS WITH RENEGING

THOMAS MASANET* ** AND
PASCAL MOYAL, * *** *Université de Lorraine and Inria PASTA*

Abstract

In this paper, we introduce a slight variation of the dominated-coupling-from-the-past (DCFTP) algorithm of Kendall, for bounded Markov chains. It is based on the control of a (typically non-monotonic) stochastic recursion by another (typically monotonic) one. We show that this algorithm is particularly suitable for stochastic matching models with bounded patience, a class of models for which the steady-state distribution of the system is in general unknown in closed form. We first show that the Markov chain of this model can easily be controlled by an infinite-server queue. We then investigate the particular case where patience times are deterministic, and this control argument may fail. In that case we resort to an ad-hoc technique that can also be seen as a control (this time, by the arrival sequence). We then compare this algorithm to the primitive coupling-from-the-past (CFTP) algorithm and to control by an infinite-server queue, and show how our perfect simulation results can be used to estimate and compare, for instance, the loss probabilities of various systems in equilibrium.

Keywords: Markov chains; stochastic matching; coupling from the past; graphs

2020 Mathematics Subject Classification: Primary 60J10
Secondary 05C70; 68M20

1. Introduction

The study of stochastic matching models is currently a very active line of research in applied probability. It has been demonstrated in various contexts that these stochastic models are suitable for capturing the dynamics of a wide range of real-time random systems, in which items enter the system at (possibly) random times, with a view to finding a *match*, which is identified as such in accordance with specified compatibility rules, given by a compatibility graph between classes of items. Matched couples leave the system right away, as soon as a match has been found. This is the case in various applications, such as peer-to-peer applications, job searches, public housing or college allocations, organ transplants, blood banks, car sharing, assemble-to-order systems, and so on. These models have been introduced in [15] for bipartite graphs (which are suitable for supply/demand-type applications) and arrivals by couples, as a variant of the seminal works [3, 16]. To account for a wider range of applications (e.g., dating websites, crossed kidney transplants, assemble-to-order systems, or car-sharing), they have

Received 18 February 2022; accepted 12 December 2023.
* Postal address: IECL, Faculté des Sciences et Technologies, Campus Aiguillettes, 54506 Vandœuvre-lès-Nancy.
** Email address: thomas.masanet@univ-lorraine.fr
*** Email address: pascal.moyal@univ-lorraine.fr

© The Author(s), 2024. Published by Cambridge University Press on behalf of Applied Probability Trust.

been generalized to general graphs (with simple arrivals) in [32], and then to hypergraphs in [40, 42] and graphs with self-loops, in [8].

Applications such as organ transplants are subject to very strong timing constraints: the patients waiting for a transplant have a finite lifetime in the system, and similarly, available organs are highly perishable and must be transplanted very quickly—hence the need to incorporate an impatience (or reneging) parameter into the system. More precisely, in this paper we address a general stochastic matching model, as defined in [32], in which the items have a finite (and possibly random) patience upon arrival, before the end of which they must find a match. Otherwise, they renege and leave the system forever. Matching models with impatience have recently been addressed for a bipartite model and the 'N' graph in [17] for a matching policy of the first-come-first-matched (FCFM) type, and from the point of view of stochastic optimization, for partially static policies, in [5]. On another hand, in [26], stability conditions, together with moment bounds at equilibrium, have been given for models in which some, but not all, classes of items are impatient, and the matching policy is of the max-weight class.

However, it is important to observe, first, that the exact computation of the stability regions of matching models is difficult for general graphs, and heavily depends on the matching policy; see e.g. [32, 38]. Second, the stationary distributions of the models at hand are in general unknown, and little is known about the characteristics of the steady state. In the existing literature, the models implementing the FCFM policy constitute the only exception, in which the stationary distribution is known explicitly. It can often be characterized in a product form, as is shown using dynamic reversibility arguments (see, for the various models, [1, 2, 8, 18, 37]), for models without reneging. Let us also observe the recent advances in [7, 19] concerning the invariance of stationary matching rates on the matching policy for various graphs—which shows that all matching policies have the same matching rates as FCFM.

However, in the cases of models with reneging, aside from the particular graph geometries addressed in [17], no exact results are known. Moreover, FCFM policies are clearly not always the best option in a real-time context: coming back to the case of organ transplants, other criteria must be taken into account, such as the level of emergency, equity, ages of the patient and donor, various levels of compatibility, and so on. Mimicking the various existing results in queueing theory, implementing policies of the match-the-longest-queue (ML) or earliest-deadline-first (EDF) type may be profitable to minimize loss, and it is significant that EDF does *not* amount to FCFM if the patience times are random.

Our aim is to analyze matching models with reneging in steady state, for general matching policies. In view of the above discussion, we thus need to assess the stationary distribution of the matching model at hand, without knowledge of this distribution in closed form. As is well known, this task can be handled through *perfect simulation* of this steady state.

Perfect simulation has been a constantly active line of research in the analysis of stochastic systems, since the pioneering works of Propp and Wilson [41] and Borovkov and Foss [11, 12]. The underlying idea is now well known: consider a discrete-event stochastic system whose stationary distribution is intractable mathematically. Then we can study the system in steady state by *precisely* simulating samples of the stationary distribution, even though the latter is not known in closed form, instead of approximating it by long-run trajectories. Then various average performance parameters at equilibrium can be assessed by Monte Carlo techniques.

The celebrated algorithm of Propp and Wilson [41] is based on coupling from the past (CFTP); namely, all trajectories of the considered Markov chain coalesce before time 0, whenever these trajectories are initiated from all possible states of the chain, far away enough in the past. This phenomenon is closely related to the concept of strong backwards coupling (see e.g.

[13] and Chapter 2.5 of [6]), and the connections between the two notions are investigated for various cases of stochastic recursions in [22]. Strong backwards coupling is the pillar of the construction of the stationary state under general non-Markov assumptions, via the use of renovating events; see e.g. [11, 12]. It is also a tool for constructing stationary states on enriched probability spaces via skew-product constructions; see [4, 31, 36].

As they rely on the exact coalescence of a family of Markov chains, CFTP algorithms are typically adapted to finite state spaces and to monotonic dynamics, using envelope techniques. Various authors have extended these settings: generalizing the ideas in [22], it is proven in [28] that geometrically ergodic Markov chains admit a CFTP algorithm of the envelope type, even if they are not monotonic, a result that was then generalized to a wider class of ergodic Markov chains in [20]. Various related approaches have since been proposed, all of which rely on the intuitive idea of simulating from the past a 'simpler' recursion, then deducing the stationary state of the recursion of interest by comparison. This is the core idea of dominated coupling from the past (DCFTP), introduced in [29, 30] and then [28]; of the bounding chains of Huber [24, 25], which are particularly adapted to mechanical-statistical contexts; and of various envelope techniques for queueing systems (see e.g. [14]). More recently, DCFTP-related methods have been implemented, together with saturation techniques, to perfectly simulate non-Markov queueing systems; see [9] for infinite-server and loss queues, and [10] for multiple-server queues.

This paper is a first contribution on the perfect sampling of stochastic matching models. We introduce two perfect sampling algorithms, Algorithms 2 and 3 below, that produce samples of the stationary distribution of stochastic matching models with reneging, in the case where arrival times are discrete. The first algorithm simply relies on the control of the model at hand by an infinite-server queue, an algorithm that would clearly not be optimal in a context with heavy traffic. Indeed, as was observed in [9], as it relies on the depletion of a corresponding infinite-server model, the coalescence time for Algorithm 2 grows exponentially as a function of the arrival rates; see [27]. Our second algorithm, Algorithm 3, is peculiar to the case where patience times are deterministic (and so the matching policies FCFM and edf are equivalent). In that case, we propose an ad-hoc control of the system simulated backwards in time by the input of the system. Then the algorithm substantially reduces the number of operations compared to the primitive CFTP. In particular, if latency is allowed, we show that Algorithm 3 also outperforms the algorithm based on control by the infinite-server queue, Algorithm 2. In fact, both Algorithms 2 and 3 can be seen as particular cases of a more general perfect sampling algorithm for bounded Markov chains, namely Algorithm 1 below, which we call perfect sampling *by control*, a condition that is closely related to those under which a DCFTP-type algorithm can be implemented.

This paper is organized as follows. After some preliminaries in Section 2, we introduce our general algorithm for perfect sampling by control in Section 3. In Section 4 we introduce the general stochastic matching model with reneging, and we introduce the two corresponding perfect sampling algorithms in Subsections 4.2 and 4.3.3. The performance of the latter algorithm is investigated in Subsection 4.3.4. We compare the performance of Algorithm 3 to that of the primitive CFTP algorithm in Subsection 4.5, and to that of Algorithm 2 in Subsection 4.4, for a model with reneging and latency. In Subsection 4.5 we provide a first application to the comparison of the steady-state performances of two matching policies (here, edf—or, in other words, FCFM—and ML).

2. Preliminaries

In what follows, $\mathbb{R}$, $\mathbb{N}$, $\mathbb{N}^*$, and $\mathbb{Z}$ denote the sets of real numbers, non-negative integers, strictly positive integers, and integers, respectively. For any two elements $a, b \in \mathbb{Z}$, let $[\![a, b]\!]$ denote the integer interval $[a, b] \cap \mathbb{Z}$.

Any (simple, finite, and undirected) graph G is denoted by $G = (\mathbb{V}, E)$, where $\mathbb{V}$ is the set of nodes and E is the set of edges for a node $i \in \mathbb{V}$. For $n \in \mathbb{N}^*$, we say that G is of size n if the cardinality $|\mathbb{V}|$ of $\mathbb{V}$ is n. For any nodes $i, j \in \mathbb{V}$, we write $i - j$ if i and j share an edge in G, that is, $\{i, j\} \in E$. Otherwise, we write $i \not- j$. For any set $U \subset \mathbb{V}$, we denote by $E(U)$ the neighborhood of U, namely,

$$E(U) = \{j \in \mathbb{V} : i - j \text{ for some } i \in U\}.$$

For simplicity, for all $i \in \mathbb{V}$ we set $E(i) := E(\{i\})$, the set of neighbors of node i in $\mathbb{V}$.

Throughout the paper, all the random variables we consider are defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

Definition 1. Let $\mathbb{X}$ and $\mathbb{V}$ be two separable metric spaces. Let $k \in \mathbb{Z}$ and $x \in \mathbb{X}$. Let f be a measurable mapping from $\mathbb{X} \times \mathbb{V}$ to $\mathbb{X}$, and let $(v_n)_{n\in\mathbb{Z}}$ be an identically distributed sequence of $\mathbb{V}$-valued random variables. We denote by $\left(X_n^k(x)\right)_{n\geq k}$ the stochastic recursive sequence (SRS) driven by $\left(f, (v_n)_{n\in\mathbb{Z}}\right)$, with initial value x at time k. Namely, $\left(X_n^k(x)\right)_{n\geq k}$ is fully determined by the recurrence equation

$$\begin{cases} X_k^k(x) & = x, \\ X_{n+1}^k(x) & = f(X_n^k(x), v_n) \quad \text{a.s. for all } n \geq k. \end{cases}$$

It is immediate that $\left(X_n^k(x)\right)_{n\in\mathbb{Z}}$ is an $\mathbb{X}$-valued Markov chain whenever the sequence $(v_n)_{n\in\mathbb{Z}}$ is independent and identically distributed (i.i.d.). Conversely, any $\mathbb{X}$-valued Markov chain $(Z_n)_{n\in\mathbb{N}}$ of fixed starting time k and initial value x, and with transition matrix Q over $\mathbb{X}$, can be represented by the SRS driven by $\left(f, (v_n)_{n\in\mathbb{Z}}\right)$, where $(v_n)_{n\in\mathbb{Z}}$ is an i.i.d. sequence of uniformly distributed random variables on [0, 1], and f is piecewise constant and satisfies, for all $x_1, x_2 \in \mathbb{X}$,

$$\lambda\left(\{x :, f(x_1, x) = x_2\}\right) = Q(x_1, x_2),$$

for λ the Lebesgue measure; see e.g. Section 2.5.3 of [6].

Fix an SRS $X := \left(X_n^k(x)\right)_{n\geq k}$. Then for all $e \in \mathbb{X}$, we set

$$\tau_e^{X,k}(x) = \inf\left\{n \geq k : X_n^k(x) = e\right\},$$

the hitting time of the value e by $\left(X_n^k(x)\right)_{n\geq k}$. If $(v_n)_{n\in\mathbb{Z}}$ is i.i.d., then $\left(X_n^k(x)\right)_{n\geq k}$ is a Markov chain, and the distribution of $\tau_e^{X,k}(x) - k$ is independent of k. In that case, we then denote by $\tau_e^X(x)$ a generic random variable that is so distributed.

3. A perfect sampling algorithm by control

In this section we present a perfect simulation algorithm, Algorithm 3, for processes that are bounded, in a sense that we will make precise hereafter.

```
Data: $a_1, \dots, a_r \in \mathbb{X}$, $b_1, \dots, b_r$, $y \in \mathbb{Y}$, a probability distribution $\mu$ on $\mathbb{V}$
$T_{down} \leftarrow -1$ ;
$T_{up} \leftarrow -1$ ; /* We initialize the starting time.                                    */
$Y \leftarrow y$;
while $Y \notin \{b_1, \dots, b_r\}$ do
    $i \leftarrow T_{up}$ ;
    $Y \leftarrow y$;
    for $j \leftarrow T_{up}$ to $T_{down}$ do
        draw $v_j$ from $\mu$; /* We draw the random variables still needed for
        this iteration                                                              */
    end
    while $i < 0$ and $Y \notin \{b_1, \dots, b_r\}$ do
        $Y \leftarrow g(Y, v_i)$ ;
        $i \leftarrow i + 1$ ;
    end
    $T_{down} \leftarrow T_{up} - 1$ ;
    $T_{up} \leftarrow 2T_{up}$
end
for $k \leftarrow 1$ to $q$ do
    if $Y = b_k$ then
        $X \leftarrow a_k$ ; /* We assign to $X$ the state corresponding to the endpoint
         reached by $Y$.                                                            */
    end
end
/* We now transition $X$ to time 0.                                                 */
while $i < 0$ do
    $X \leftarrow f(X, v_i)$ ;
    $i \leftarrow i + 1$ ;
end
return $X$
```

Algorithm 1: Simulation of the stationary probability of X.

Our procedure is closely related to that of the DCFTP algorithm introduced by Kendall and Moller in [29, 30]. Algorithm 1 proceeds roughly as follows: we simulate from the past an auxiliary chain Y, until it has reached one of the *endpoints*, at which time we start simulating a trajectory of the CTMC X, up to time 0. As will be shown hereafter, under certain conditions the output of Algorithm 1 is sampled exactly from the stationary distribution of X.

Until the end of this section, we fix three separable metric spaces $\mathbb{X}$, $\mathbb{Y}$, and $\mathbb{V}$, and two mappings $f : \mathbb{X} \times \mathbb{V} \to \mathbb{X}$ and $g : \mathbb{Y} \times \mathbb{V} \to \mathbb{Y}$.

3.1. A control condition

The *control* of an SRS of interest by an auxiliary one is the key to our perfect simulation algorithm. It is defined below.

Definition 2. Let $(X_n)_{n\in\mathbb{Z}}$ and $(Y_n)_{n\in\mathbb{Z}}$ be two SRSs, respectively valued in $\mathbb{X}$ and $\mathbb{Y}$, and let $r \in \mathbb{N}^*$. We say that $(Y_n)_{n\in\mathbb{Z}}$ *r-controls* $(X_n)_{n\in\mathbb{Z}}$ if there exist $b_1, \ldots, b_r, y \in \mathbb{Y}$ and $a_1, \ldots, a_r \in \mathbb{X}$ such that the following holds:

$$\text{For all } i \in [\![1, r]\!],\ k \in \mathbb{Z},\ \text{and } n \geq k, \quad \left[Y_n^k(y) = b_i\right] \Longrightarrow \left[\forall x \in \mathbb{X},\ X_n^k(x) = a_i\right]. \tag{1}$$

The points $b_1, \ldots, b_r$ are called the *endpoints* of Y. If $r = 1$, we simply say that Y *controls* X.

The following result establishes that under certain conditions including the control of $(X_n)_{n\in\mathbb{Z}}$ and $(Y_n)_{n\in\mathbb{Z}}$, Algorithm 1 terminates almost surely, and the output is a sample of the stationary distribution of the SRS $(X_n)_{n\in\mathbb{Z}}$.

Theorem 1. *Suppose that the sequence $(v_n)_{n\in\mathbb{Z}}$ is i.i.d., and let X and Y be two SRSs respectively driven by $\big(f, (v_n)_{n\in\mathbb{Z}}\big)$ and $\big(g, (v_n)_{n\in\mathbb{Z}}\big)$. Suppose that X is r-controlled by Y for $b_1, \ldots, b_r, y$ and $a_1, \ldots, a_r$, and that it holds that*

$$\mathbb{P}\left[\tau_{b_i}^Y(y) < \infty\right] = 1, \qquad i \in [\![1, r]\!]. \tag{2}$$

Then Algorithm 1 terminates almost surely, and its output is sampled from the unique stationary distribution of X.

Proof. We first show that Algorithm 1 terminates almost surely. To see this, observe that for any $i \in [\![1, r]\!]$ and any $N \in \mathbb{N}$,

$$\begin{aligned}\mathbb{P}\left(\bigcup_{n\in\mathbb{N}}\left\{\tau_{b_i}^{Y,-2^n}(y) \leq 0\right\}\right) &= \mathbb{P}\left(\bigcup_{n\in\mathbb{N}}\left\{\tau_{b_i}^{Y,-2^n}(y) + 2^n \leq 2^n\right\}\right)\\ &\geq \mathbb{P}\left(\left\{\tau_{b_i}^{Y,-2^N}(y) + 2^N \leq 2^N\right\}\right) = \mathbb{P}\left(\left\{\tau_{b_i}^{Y}(y) \leq 2^N\right\}\right),\end{aligned}$$

in view of the stationarity of the input. So we obtain that

$$\begin{aligned}\mathbb{P}\left(\bigcup_{n\in\mathbb{N}}\left\{\tau_{b_i}^{Y,-2^n}(y) \leq 0\right\}\right) &\geq \lim_{N\to+\infty} \mathbb{P}\left(\left\{\tau_{b_i}^{Y}(y) \leq 2^N\right\}\right)\\ &= \mathbb{P}(\{\tau_{b_i}^Y(y) < +\infty\}) = 1,\end{aligned}$$

showing that Algorithm 1 terminates almost surely.

Now, let N be the backwards coalescence time of the chain X, that is, the smallest starting time for which the CFTP algorithm terminates for X; in other words,

$$N = \min\left\{n \geq 0 : X_0^{-n}(x) = X_0^{-n}(x') \text{ for all } x \neq x' \in \mathbb{X}\right\}. \tag{3}$$

Let R be the smallest termination time of Algorithm 1. Then, by the very definition of Algorithm 1 and (1), there exist $i \in [\![1, q]\!]$ and a time $n_0 > 0$ such that $-R < -n_0$, and such that $X_{-n_0}^{-R}(x) = X_{-n_0}^{-R}(x') = a_i$ for all $x, x' \in \mathbb{X}, x \neq x'$, so that

$$X_0^{-R}(x) = X_0^{-R}(x') = X_0^{-n_0}(a_i), \text{ for all } x, x' \in \mathbb{X}, x \neq x'. \tag{4}$$

In particular, it naturally follows from (3) that we necessarily have $R \geq N$; otherwise all versions of the chain X starting from a time posterior to $-N$ would have coalesced before 0, an absurdity. In particular, N is almost surely finite. In the terms of [22], the vertical backwards coalescence time of X is successful, and so it follows from Theorem 4.1 in [22], first, that there exists a unique invariant probability π for the chain X, and second, that the output $X_0^{-N}(x)$ of the CFTP algorithm when started from any $x \in \mathbb{X}$ is sampled from π. But it also follows from (4) that

$$X_0^{-R}(x) = X_0^{-N}(x) = X_0^{-n_0}(a_i), \qquad \text{for all } x \in \mathbb{X}.$$

So Algorithm 1 and the CFTP algorithm produce the same output, which completes the proof. □

Remark 1. The assumptions of Theorem 1 are satisfied in particular if Y is positive recurrent and irreducible on the discrete state space $\mathbb{Y}$, or if the distribution of Y has atoms at points $b_1, \ldots, b_r$ with finite hitting times from y.

Remark 2. It follows from the equivalence shown in Theorem 4.2 of [22] that under the assumptions of Theorem 1, the Markov chain X is uniformly ergodic, since the vertical coalescence time for X is successful.

3.2. Renovating events and small sets

Assumption (1) is key to our analysis. Under this control condition, the value of the SRS Y forces that of X at time n, whatever the value of X at time N. This is reminiscent of the concept of a renovating event, as introduced by Borovkov and Foss; see [11, 12].

Definition 3. Let us recall the following. Let X be an SRS driven by f and $(v_n)_{n\in\mathbb{Z}}$, and let $(A_n)_{n\in\mathbb{N}}$ be a sequence of events. We say that $(A_n)_{n\in\mathbb{N}}$ is a sequence of *renovating events* of length m and associated mapping $h : \mathbb{V}^m \to \mathbb{X}$ for the chain X if for any $n \in \mathbb{Z}$, on A_n we have

$$X_{n+m} = h(v_n, \ldots, v_{n+m-1}).$$

Now suppose that (1) holds for $a_1, \ldots, a_r, b_1, \ldots, b_r$ and y. Then it is easily seen that for all $k \in \mathbb{Z}$, $i \in [\![1, r]\!]$, and $x \in \mathbb{X}$, $\left(\{Y_n^k(y) = b_i\}\right)_{n\geq k}$ is a sequence of renovating events of length 1 for the sequence $\left(X_n^k(x)\right)_{n\geq k}$. Indeed, for all $n \geq k$, on $\{Y_n^k(y) = b_i\}$ we get that $X_n^k(x) = a_i$, and therefore

$$X_{n+1}^k(x) = f(a_i, v_n) =: h(v_n).$$

One can then give various conditions on the events $\left(\{Y_n^k(y) = b_i\}\right)_{n\geq k}$ which imply that there exists a stationary version of the chain $(X_n)_{n\in\mathbb{N}}$; see e.g. Theorem 2.5.3 and Property 2.5.5 in [6].

There is also an insightful connection between the control condition of Definition 2 and the concept of a small set, which is recalled below as formulated in Chapter 5 of [33].

Definition 4. For a positive integer m, we say that the subset $A \subset \mathbb{X}$ is m- *small* for the Markov chain $(X_n)_{n\in\mathbb{N}}$ on $\mathbb{X}$ if there exist $\eta_m > 0$ and a non-null Borel measure μ_m on $\mathbb{X}$ such that

$$\forall x \in A,\ \forall B \in \mathcal{B}(\mathbb{X}), \qquad \mathbb{P}\left[X_m \in B \mid X_0 = x\right] \geq \eta_m \mu_m(B).$$

Thus, starting from such a set, the chain (partially) regenerates in a finite horizon of size m, since after that, the transitions of the chain do not depend on the starting point x with strictly positive probability. The existence of small sets is of crucial use in the construction of uniformly ergodic Markov chains; see [20, 22, 28].

It is significant that under the control condition of Definition 2 and (2), the whole set $\mathbb{X}$ is small. To see this, observe that for any $i \in [\![1, r]\!]$, for any $m \in \mathbb{N}$ such that $\mathbb{P}\left[\tau^Y_{b_i}(y) = m-1\right] > 0$, in view of the Markov property, for all $x \in \mathbb{X}$ and all Borel subsets $B \subset \mathbb{X}$ we have

$$\begin{aligned}
\mathbb{P}\left[X_m \in B \mid X_0 = x\right] &\geq \mathbb{P}\left[\{X_m \in B\} \cap \{X_{m-1} = a_i\} \mid X_0 = x\right] \\
&= \mathbb{P}\left[X_{m-1} = a_i \mid X_0 = x\right] \mathbb{P}\left[X_m \in B \mid \{X_{m-1} = a_i\} \cap \{X_0 = x\}\right] \\
&\geq \mathbb{P}\left[\tau^Y_{b_i}(y) = m-1\right] \mathbb{P}\left[X_1 \in B \mid X_0 = a_i\right] \\
&=: \delta^i_m \mu^i_m(B).
\end{aligned}$$

Hence $\mathbb{X}$ is m-small.

3.3. The ordered case

A typical context in which the control of the SRS X by the SRS Y occurs is when the two sequences are constructed on the same input, and their driving maps satisfy some monotonicity properties, which we detail below. Throughout this section, $(\mathbb{U}, \prec)$ denotes a partially ordered space, and we define two mappings $\varphi : \mathbb{X} \longmapsto \mathbb{U}$ and $\psi : \mathbb{Y} \longmapsto \mathbb{U}$.

Definition 5. We say that the mapping $f : \mathbb{X} \to \mathbb{X}$ is dominated (for $\mathbb{U}$, φ, and ψ) by the mapping $g : \mathbb{Y} \to \mathbb{Y}$, and write $f \prec^{\mathbb{U},\varphi,\psi} g$, if

$$\forall x \in \mathbb{X},\ y \in \mathbb{Y}, \quad \left[\varphi(x) \prec \psi(y)\right] \Longrightarrow \left[\varphi \circ f(x) \prec \psi \circ g(y)\right].$$

In the definition above, $\mathbb{U}$ is an auxiliary partially ordered set that is used for comparing f to g via the projections φ and ψ. Observe the following simple special case.

Proposition 1. *In the case where $\mathbb{X} = \mathbb{Y} = \mathbb{U}$, $\mathbb{X}$ is partially ordered by $\prec$, and $\varphi = \psi = \mathrm{i}$ is the identity function, we have $f \prec^{\mathbb{X},\mathrm{i},\mathrm{i}} g$ under either one of the conditions below:*

(i) g is $\prec$-non-decreasing and pointwise lower-bounded by f;

(ii) f is $\prec$-non-decreasing and pointwise upper-bounded by g.

Proof. Plainly, for all $x, y \in \mathbb{X}$ such that $x \prec y$, if we assume that (i) holds, then we get $f(x) \prec g(x) \prec g(y)$, whereas if (ii) holds we obtain that $f(x) \prec f(y) \prec g(y)$. □

Proposition 2. *Let X and Y be two SRSs respectively driven by $(f, (v_n)_{n\in\mathbb{Z}})$ and $(g, (v_n)_{n\in\mathbb{Z}})$, where the input $(v_n)_{n\in\mathbb{Z}}$ is i.i.d. on $\mathbb{V}$. Suppose that $f(., v) \prec^{\mathbb{U},\varphi,\psi} g(., v)$ for all $v \in \mathbb{V}$, where $\mathbb{U}$ admits the $\prec$-minimal point o. Suppose also that $\varphi^{-1}(o) = \{a\}$, that there exists $y \in \mathbb{Y}$ such that*

$$\forall x \in \mathbb{X}, \quad \varphi(x) \prec \psi(y), \tag{5}$$

and that $\tau_b^Y(y)$ is almost surely finite for some $b \in \psi^{-1}(o)$. Then Algorithm 1 for y, a, and b terminates almost surely and produces a sample of the unique stationary distribution of X.

Proof. We aim to show that Y controls X for b, y, a. Let $k, n \in \mathbb{Z}$ be such that $n > k$ and $Y_n^k(y) = b$. Let $x \in \mathbb{X}$. We show by induction on ℓ that for all $\ell \in [\![k, n]\!]$,

$$\varphi(X_\ell^k(x)) \prec \psi(Y_\ell^k(y)). \tag{6}$$

First, from (5) we get that $\varphi(X_k^k(x)) = \varphi(x) \prec \psi(y) = \psi(Y_k^k(y))$, so (6) holds for $\ell = k$. Suppose that it is true at rank $\ell \in [\![k, n-1]\!]$, i.e., that $\varphi(X_\ell^k(x)) \prec \psi(Y_\ell^k(y))$. Then, from the assumption that g dominates f, we obtain that

$$\varphi(X_{\ell+1}^k(x)) = \varphi(f(X_\ell^k(x), v_\ell)) \prec \psi(g(Y_\ell^k(y), v_\ell)) = \psi(Y_{\ell+1}^k(y)),$$

so (6) holds at rank $\ell + 1$. It is therefore true for all $\ell \in [\![k, n]\!]$. In particular, we have that

$$\varphi(X_n^k(x)) \prec \psi(Y_n^k(y)) = \psi(b) = o,$$

implying that $X_n^k(x) = a$. Thus Y controls X, and we conclude using Theorem 1. □

As a conclusion, provided that $f \prec^{\mathbb{U},\varphi,\psi} g$, Algorithm 1 provides a perfect sampling algorithm for the SRS X. In fact, in this ordered case, Algorithm 1 is closely related to the DCFTP algorithms of Kendall; see [20, 28]. Specifically, as in [20], thanks to (5) we have an upper-bound process Y that we can simulate backwards in time. We also have a lower-bound process, namely the constant process equal to b. Similarly to the sandwiching method in [41], we only have to simulate the process Y starting at state y. When that process meets the lower bound backwards in time, it means that coalescence has been detected. Then, as in [20, 28], we simulate X starting from a single state until time 0. Notice that the DCFTP algorithms introduced in [20, 28] use geometric ergodicity and are based on the construction of small sets. As we observed above (Section 3.2), the control condition implies the smallness of $\mathbb{X}$, so our approach is reminiscent of this idea.

Observe that similar approaches are used for the perfect sampling of loss queueing systems in [9], which uses the domination of the system by an infinite-server queue in some sense (an idea that we also use in the construction of Section 4.2 below), and likewise for the perfect sampling of multiple-server queues in [10].

Remark 3. The above DCFTP conditions are in fact reminiscent of stochastic domination conditions for the construction of stationary SRSs in the general stationary ergodic context. For instance, for $\mathbb{X} = E$ a lattice space, Condition (i) in Proposition 1 above amounts to Condition (H1) in [36] for any SRSs X and Y that are respectively driven by f and g, and a common input $(v_n)_{n\in\mathbb{Z}}$. This latter condition guarantees, under general stationary ergodic assumptions, the existence of a stationary version of the SRS X, at least on an extended probability space, provided that a stationary version of the SRS Y exists on the original one. See [31] and Theorem 3 in [36].

4. A stochastic matching model with impatience

In this section we address the perfect sampling of the stationary state of a class of models, which we refer to as 'general stochastic matching models with impatience'.

4.1. The model

We consider a general stochastic matching model (GM), as defined in [32]: items enter a system one by one, and each of them belongs to a determinate class. The set of classes is denoted by $\mathbb{V}$ and is identified with $[\![1, |\mathbb{V}|]\!]$. We fix a simple, connected graph $G = (\mathbb{V}, E)$ having set of nodes $\mathbb{V}$, termed a *compatibility graph*. Upon the arrival of an incoming item of class, say, $i \in \mathbb{V}$, either it is matched with an item present in the buffer, of a class j such that i shares an edge with j in G, if any, or if no such item is available, it is stored in the buffer to wait for a match. Whenever several possible matches are possible for an incoming item i, a *matching policy* determines what the match of i is, without ambiguity. Each matched pair departs the system right away.

A GM model with impatience is a GM model in which each item entering the system is assigned a patience time upon arrival. If the item under consideration has not been matched at the end of its patience time, then it leaves the system forever. To formalize this, after fixing the compatibility graph $G = (\mathbb{V}, E)$ and the matching policy Φ, we assume that arrivals occur at integer times, i.e., we suppose that the generic inter-arrival time ξ is constant equal to one, and fix two i.i.d. sequences $(V_n)_{n\in\mathbb{Z}}$ and $(P_n)_{n\in\mathbb{Z}}$, where for all $n \in \mathbb{Z}$, $P_n \in \mathbb{R}_+$ and $V_n \in \mathbb{V}$ respectively represent the patience time and the class of the nth item entering the system. By V and P we denote generic random variables distributed like $(V_n)_{n\in\mathbb{Z}}$ and $(P_n)_{n\in\mathbb{Z}}$, respectively, and we assume throughout that the random variable P is integrable. The two sequences $(V_n)_{n\in\mathbb{Z}}$ and $(P_n)_{n\in\mathbb{Z}}$ are not necessarily independent. In particular, it can be the case that the patience time P_n of the nth item depends on its class V_n. In what follows, we denote by μ the law of V on $\mathbb{V}$.

The class of models defined in Section 4.1 admits the following Markov representation. Define the set

$$\mathbb{X} := \{\emptyset\} \cup \bigcup_{q=1}^{\infty} \left(\mathbb{R}_+^* \times \mathbb{V}\right)^q .$$

For all $t \geq 0$, let $Q(t)$ be the number of customers in the system at time t, and let us define the *profile* of the system at t as the following element of $\mathbb{X}$:

$$X(t) = \begin{cases} \left(\left(R^1(t), V^1(t)\right), \cdots, \left(R^{Q(t)}(t), V^{Q(t)}(t)\right)\right) & \text{if } Q(t) \geq 1, \\ \emptyset & \text{otherwise,} \end{cases} \tag{7}$$

where for all $i \in [\![1, Q(t)]\!]$, we denote by $R^i(t)$ (resp., $\mathbb{V}^i(t)$) the remaining patience (resp., the class) at time t of the ith item in line at time t, in the order of arrival. If the system is empty at t, we again set $X(t) = \emptyset$.

Definition 6. We say that the matching policy Φ is *admissible* if, upon each arrival, the choice of the match amongst compatible items in line at t, if any, is made solely according to the knowledge of *X(t)*, and possibly of a draw that is independent of everything else.

Remark 4. It is easily seen that matching policies that depend only on the arrival times (first-come-first-matched (FCFM) or last-come-first-matched (lcfm)), remaining patience times (earliest-deadline-first, latest-deadline-first), matching policies that depend on the queue sizes of the various nodes (match-the-longest, match-the-shortest, max-weight), and priority policies are all admissible. See e.g. [26, 32, 37] for a detailed presentation of admissible policies for classical matching models.

Set $(T_n)_{n\in\mathbb{Z}} = (n)_{n\in\mathbb{Z}}$, the arrival times to the system, and for all $n \in \mathbb{Z}$, denote by $X_n = X(T_n^-) = X(n^-)$ the state of the system seen by the customer entering at time n. Then we obtain the following result.

Proposition 3. *For any admissible matching policy Φ, the profile sequence $(X_n)_{n\in\mathbb{Z}}$ is stochastic recursive, driven by the sequence of pairs $((V_n, P_n))_{n\in\mathbb{Z}}$ and a mapping $f^\Phi : \mathbb{X} \times (\mathbb{R}_+ \times \mathbb{V}) \longmapsto \mathbb{X}$ that depends on Φ and possibly on a random draw independent of everything else. In other words, we get that*

$$X_{n+1} = f^\Phi\left(X_n, (P_n, V_n)\right), \quad n \in \mathbb{Z}.$$

Proof. The construction of f^Φ is immediate: if the incoming item at n is matched upon arrival, the pair corresponding to this match, determined by Φ, is erased from the vector X_n; otherwise, the pair (V_n, P_n) is added at the end of the vector X_n. Lastly, the pairs (possibly including the incoming pair (V_n, P_n)) whose second coordinate is strictly less than 1 at n are erased from the vector X_n (because they will have reneged by time $n + 1$), and the second coordinates of all other pairs in X_n, if any, decrease by 1. □

4.2. A first perfect sampling algorithm

We can now design a first perfect sampling algorithm for matching models with impatience, which is simply based on control (in the sense of Section 3) by an infinite-server system. In this context, we let $Y := (Y_n)_{n\in\mathbb{Z}}$ be an $\mathbb{R}_+$-valued SRS defined by the recursion

$$Y_{n+1} = [\max(Y_n, P_n) - 1]^+ =: g(Y_n, P_n), \quad n \in \mathbb{Z}. \tag{8}$$

Then, for all n, Y_n can be interpreted as the largest remaining service time of a D/GI/∞ queue of service times $(P_n)_{n\in\mathbb{Z}}$, upon the arrival of the nth customer. As the generic random variable P is assumed integrable, it is well known that whenever

$$\mathbb{P}(P \le \xi) = \mathbb{P}(P \le 1) > 0, \tag{9}$$

the Markov chain $(Y_n)_{n\in\mathbb{Z}}$ is positive recurrent; see e.g. Corollary 4.32 in [21, 43], and the generalization to the case where $(P_n)_{n\in\mathbb{Z}}$ is stationary ergodic, combining Lemma 5 of [34] with Corollary 2 in [35]. Consider Algorithm 2, which is a declination of Algorithm 1 started with $y = m$ for m defined below, for Y the recursion defined by (8), $q = 1$, $a_1 = \emptyset$, and $b_1 = 0$.

Theorem 2. *Under Condition (9), the profile Markov chain $(X_n)_{n\in\mathbb{Z}}$ admits a unique stationary distribution. If moreover there exists $m > 0$ such that $\mathbb{P}(P \le m) = 1$, then Algorithm 2 terminates almost surely, and its output is sampled from the stationary distribution of $(X_n)_{n\in\mathbb{Z}}$.*

Proof. We apply Proposition 2, by setting in this case

$$\begin{aligned} \varphi: \quad & \mathbb{X} && \longrightarrow \mathbb{R}_+ \\ & x = \big((r_1, v_1), \cdots, (r_q, v_q)\big) \neq \emptyset && \longmapsto \max\{r_i : i \in [\![1, q]\!]\} \\ & \emptyset && \longmapsto 0. \end{aligned} \tag{10}$$

As the time spent in the system by any item is less than or equal to its patience time, for any $n \in \mathbb{Z}$, $\varphi(X_n)$ corresponds to the largest remaining maximal sojourn time in the system of an

```
Data: A probability distribution $\mu$ on $\mathbb{V} \times \mathbb{R}^+$
$T_{down} \leftarrow -1$ ;
$T_{up} \leftarrow -1$ ; /* We initialize the starting time.                         */
$Y \leftarrow m$ ;
while $Y \neq 0$ do
    $i \leftarrow T_{up}$ ;
    $Y \leftarrow \varnothing$ ;
    for $j \leftarrow T_{up}$ to $T_{down}$ do
        draw $(v_j, p_j)$ from $\mu$; /* We draw the random variables still needed at
        this iteration                                                             */
    end
    while $i < 0$ and $Y \neq 0$ do
        $Y \leftarrow [\max(Y, p_i) - 1]^+$ ;
        $i \leftarrow i + 1$ ;
    end
    $T_{down} \leftarrow T_{up} - 1$ ;
    $T_{up} \leftarrow 2T_{up}$ ;
end
$X \leftarrow \varnothing$ ; /* As $Y$ has reached $0$ we know that $X$ has reached $\varnothing$.      */
/* We now transition $X$ to time $0$ as a matching system.                      */
while $i < 0$ do
    $X \leftarrow f^{\Phi}(X, (v_i, p_i))$ ;
    $i \leftarrow i + 1$ ;
end
return $X$
```

Algorithm 2: Simulation of the stationary probability of X - Matching model with impatience.

item in the system just before time T_n. Consequently, for any $(p, v) \in \mathbb{R}_+ \times \mathbb{V}$, for all $x \in \mathbb{X}$ we obtain that

$$\varphi\left(f^{\Phi}(x, (p, v))\right) \leq \left[\max(\varphi(x), p) - 1\right]^+ = g(\varphi(x), p). \tag{11}$$

Therefore, for any $x \in \mathbb{X}$ and $y \in \mathbb{R}_+$ such that $\varphi(x) \leq y$, for any (p, v), as $g(., p)$ is non-decreasing on $\mathbb{R}_+$ we get that

$$\varphi\left(f^{\Phi}(x, (p, v))\right) \leq g(y, p).$$

Proposition 2 completes the proof. □

4.3. Deterministic patience times

Whenever Condition (9) does not hold, the existence and uniqueness of a stationary distribution for the Markov chain $(X_n)_{n\in\mathbb{N}}$ are not granted. One then has to resort to ad-hoc techniques to show stability and to sample the stationary state.

In this section, we consider the particular case of the previous model, in which patience times are deterministic. Specifically, we suppose that $P \equiv p + \varepsilon$, for some $p \in \mathbb{N}^*$ and $0 < \varepsilon < 1$. Assuming that patience times are not integers, while arrivals occur at integer times, allows us to avoid the ambiguous situation in which an item enters the system and finds an item with remaining patience time zero. In practice, any incoming item can be matched either upon arrival, or with one of the p items that enter subsequently. If not, the item is lost before the arrival of the $(p+1)$th item after it, because its remaining time then equals $\varepsilon - 1 < 0$. For short, we denote such a matching model by (G, Φ, μ, p).

Clearly, in this context, (9) fails. (Notice that taking $p = 0$ in the present construction would lead to a system in which no item could ever be matched.) In this section, we show that such systems are nevertheless positive recurrent, and construct an alternative perfect sampling algorithm that is another declination of Algorithm 3, and is again based on the control condition defined in Section 3.

4.3.1. *Alternative Markov representation.* In this particular case, the profile Markov chain can be simplified so as to yield the following alternative, simpler Markov representation of the system state.

Definition 7. For all $n \in \mathbb{Z}$, the *word-profile* of the system just before time n is defined by the word

$$\tilde{X}_n = w_1 \cdots w_p \in (\mathbb{V} \cup \{0\})^p,$$

where for all $i \in [\![1, p]\!]$,

$$w_i = \begin{cases} V_{n-p+i-1} & \text{if the item entered at } n-p+i-1 \text{ was not matched before } n, \\ 0 & \text{otherwise.} \end{cases}.$$

In particular, if the item that entered at time $n - p$ is still in the system at time n (its class thus appearing as the first letter of the word $\tilde{X}_n$), it is either matched with the incoming item at time n, or considered lost.

We call $\tilde{\mathbb{X}} \subset (\mathbb{V} \cup \{0\})^p$ the (finite) state space of $\tilde{X}$. Similarly to Proposition 3, it is immediate that for any admissible policy Φ, the sequence $\left(\tilde{X}_n\right)_{n\in\mathbb{Z}}$ is a Markov chain, and we denote by $\tilde{f}^\Phi$ the (deterministic, up to a possible draw that is independent of everything else) map $\tilde{f}^\Phi : \tilde{\mathbb{X}} \times \mathbb{V} \to \tilde{\mathbb{X}}$ such that

$$\tilde{X}_{n+1} = \tilde{f}^\Phi\left(\tilde{X}_n, V_n\right), \quad n \in \mathbb{Z}.$$

4.3.2. *Synchronizing words.* For a fixed model (G, Φ, μ, p), with $G = (V, E)$, let $\mathbb{V}^*$ be the set of words on $\mathbb{V}$. For any word $v = v_1 \cdots v_l$ in $\mathbb{V}^*$ and any $\tilde{X} \in \tilde{\mathbb{X}}$, let us denote by $W^\Phi(\tilde{X}, v) \in \tilde{\mathbb{X}}$ the state of a system started at $\tilde{X}$ and receiving the arrivals $v_1, \ldots, v_l$ in that order.

Definition 8. Fix a model (G, Φ, μ, p). A word $w = w_1 \cdots w_q \in \mathbb{V}^*$ is said to be *synchronizing* if

$$\exists z(w) \in \tilde{\mathbb{X}} : \forall \tilde{x} \in \tilde{\mathbb{X}},\ W^\Phi(\tilde{x}, w) = z(w).$$

In other words, w is a synchronizing word if all buffers synchronize to some value $z(w)$ whenever they are fed by a common arrival scenario w, whatever the initial state. It is obvious how synchronizing words can be used for perfect simulation. Indeed, if we start the Markov chain at a time $-M$ from all possible states, observing a synchronizing word of length $q < M$ amongst the arrivals (in the sense that the classes of q consecutive incoming items are given by the letters of w, in that order) clearly guarantees that all chains have coalesced by time 0. In fact, recalling Definition 4, it is immediate that whenever there exists a synchronizing word w, the whole set $\tilde{\mathbb{X}}$ is $(q+1)$-small for the chain $\left(\tilde{X}_n\right)_{n\in\mathbb{Z}}$. Indeed, for all $\tilde{x} \in \tilde{\mathbb{X}}$ and $B \subset \tilde{\mathbb{X}}$,

$$\begin{aligned}\mathbb{P}\left[\tilde{X}_{q+1} \in B \mid \tilde{X}_0 = \tilde{x}\right] &\geq \mathbb{P}\left[\{\tilde{X}_{q+1} \in B\} \cap \{V_0V_1 \cdots V_{q-1} = w\} \mid \tilde{X}_0 = \tilde{x}\right] \\ &= \mathbb{P}\left[V_0V_1 \cdots V_{q-1} = w\right] \mathbb{P}\left[\tilde{X}_{q+1} \in B \mid \{V_0V_1 \cdots V_{q-1} = w\} \cap \{\tilde{X}_0 = \tilde{x}\}\right] \\ &= \prod_{i=0}^{q-1} \mu(w_i)\mathbb{P}\left[\tilde{X}_{q+1} \in B \mid \tilde{X}_q = z\right].\end{aligned}$$

In fact, our approach hereafter for perfect simulation is reminiscent of the small-set techniques for exact sampling in [23, 39, 44]. Specifically, we will use the arrivals of synchronizing words as a control to ensure the coalescence of all versions of the Markov chains.

We first provide a sufficient condition for the existence of synchronizing words, for any discrete matching system. Hereafter, for any $k, \ell \in V$ we write k–ℓ if $(k, \ell) \in E$, that is, if the nodes k and ℓ share an edge in G. Otherwise, we write $k \not- \ell$.

Definition 9. Let $w \in \mathbb{V}^*$. We say that the word of length $2p$ given by $w = w_1 \cdots w_{2p} \in \mathbb{V}^*$ is *strongly synchronizing* if

$$\forall i \in [\![1, p]\!],\ \forall j \in [\![p+1, p+i]\!],\ w_i \not- w_j.$$

The term 'strongly synchronizing' is justified by the following result.

Theorem 3. *In a discrete matching model with impatience* (G, Φ, μ, p)*, any strongly synchronizing word is a synchronizing word.*

Proof. Let $w = w_1 \cdots w_{2p}$ be a strongly synchronizing word, and let $u = w_1 \cdots w_p$ and $v = w_{p+1} \cdots w_{2p}$. Let $\tilde{x} \in \tilde{\mathbb{X}}$, and let $\mathbf{0}_p = \underbrace{0 \cdots 0}_{p}$ be the empty state. As u is of length p, any item present in the buffer represented by $\tilde{x}$ is no longer in there after the arrivals represented by u (it is either matched or discarded, possibly just after the arrival of the last item of class w_p). Therefore $W^\Phi(\tilde{x}, u) = u' = w'_1, \ldots, w'_p$ where for all $i \in [\![1, p]\!]$, $w'_i = w_i$ if the corresponding item is still in the buffer after these arrivals, and $w'_i = 0$ otherwise. As w is strongly synchronizing, for any $i \in [\![1, p]\!]$ such that $w'_i \neq 0$ and any $j \in [\![p+1; p+i]\!]$, we have that $w'_i \not- w_j$. All items corresponding to non-zero letters of u' are not matched, because their patience necessarily expires before the arrival of a compatible item, and no letters from v can be matched to a letter in u'. Therefore, if $j, h \in [\![p+1, 2p]\!]$ are such that the item corresponding to w_j is matched to that corresponding to w_h if we add v to the empty buffer $\mathbf{0}_p$, then this is also the case if we add v to the buffer u'. In other words, we get that

$$W^\Phi(\tilde{x}, w) = W^\Phi(W^\Phi(\tilde{x}, u), v) = W^\Phi(u', v) = W^\Phi(\mathbf{0}_p, v).$$

As this is true for any $\tilde{x} \in \tilde{\mathbb{X}}$, w is a synchronizing word for $z(w) := W^\Phi(\mathbf{0}_p, v)$. □

We proceed with two technical lemmas. In what follows, for all $a \in \mathbb{V} \cup \{0\}$ and all $k \in [\![0, p]\!]$, we define the following word of length p:

$$x^a(k) = \underbrace{0 \cdots 0}_{k} \underbrace{a \cdots a}_{p-k}.$$

First observe the following.

Lemma 1. *Consider a matching model with impatience* $(G, \textsc{fcfm}, \mu, p)$*, with matching policy* FCFM*. Let* $a \in \mathbb{V}$*. Then, for all* $k \in [\![0, p-1]\!]$*, for all words* w *of length* p*,* $W^{\Phi}(x^a(k), w)$ *and* $W^{\Phi}(x^a(k+1), w)$ *differ at most by one letter in some position i (substituting 0 to the ith letter).*

Proof. Let $k \in [\![0, p-1]\!]$, and write $w = w_1 \cdots w_p$. With some abuse of notation, in the proof below, the matching procedure of the initial state $x^a(k)$ (or $x^a(k+1)$) with the arrival represented by w is itself called $W^{\text{FCFM}}(x^a(k), w)$ (or $W^{\text{FCFM}}(x^a(k+1), w)$).

If $w_i \not- a$ for all $i \in [\![1, p]\!]$, then we trivially get that

$$W^{\text{FCFM}}(x^a(k), w) = W^{\text{FCFM}}(x^a(k+1), w).$$

Otherwise, let $i_1, \ldots, i_l$ be the indices, in increasing order, of the letters of w matched with letters of $x^a(k+1)$ in $W^{\text{FCFM}}(x^a(k+1), w)$. There are three possibilities for the indices (in increasing order) of the letters of w that are matched with letters of $x^a(k)$ in $W^{\text{FCFM}}(x^a(k), w)$ (which we call for short 'the indices' in the discussion hereafter):

1. The first a of $x^a(k)$ is matched in $W^{\text{FCFM}}(x^a(k), w)$ with a letter of w of index $i_0 < i_1$. Then all the remaining a's in $x^a(k)$ are matched in $W^{\text{FCFM}}(x^a(k), w)$ exactly like the a's in $W^{\text{FCFM}}(x^a(k+1), w)$, and so the indices are precisely $i_0, i_1, \ldots, i_l$.
2. The first a of $x^a(k)$ is not matched in $W^{\text{FCFM}}(x^a(k), w)$. Then all the remaining a's of $x^a(k)$ are matched in $W^{\text{FCFM}}(x^a(k), w)$ exactly like the a's in $W^{\text{FCFM}}(x^a(k+1), w)$, and the indices are again $i_1, \ldots, i_l$.
3. The first matched a of $x^a(k)$ in $W^{\text{FCFM}}(x^a(k), w)$ is matched with the letter of index i_1 in w. Then, in $W^{\text{FCFM}}(x^a(k), w)$, either the indices of the matched letters of w are the same as in $W^{\text{FCFM}}(x^a(k+1), w)$ (and then the last a in $x^a(k)$ remains unmatched), or the first $p-k-1$ a's of $x^a(k)$ are matched with letters of w at indices $i_1, \ldots, i_l$, and the last a is matched with a letter of w of index i_{l+1}, with $i_l < i_{l+1}$, in which case the indices are $i_1, \ldots, i_{l+1}$.

If the indices are $i_1, \ldots, i_l$, then $W^{\text{FCFM}}(x^a(k), w) = W^{\text{FCFM}}(x^a(k+1), w)$. If the indices are $i_0, i_1, \ldots, i_l$ or $i_1, \ldots, i_{l+1}$, then there is a letter b of w that is not matched with an a of $x^a(k+1)$ in $W^{\text{FCFM}}(x^a(k+1), w)$, but is matched with an a of $x^a(k)$ in $W^{\text{FCFM}}(x^a(k), w)$. Then, either that letter b remains unmatched in $W^{\text{FCFM}}(x^a(k+1), w)$, in which case $W^{\text{FCFM}}(x^a(k+1), w)$ and $W^{\text{FCFM}}(x^a(k), w)$ differ only at the index i_0 (or i_{l+1}), where there is a b in $W^{\text{FCFM}}(x^a(k+1), w)$ and 0 in $W^{\text{FCFM}}(x^a(k), w)$; or b is matched with a letter c of w in $W^{\text{FCFM}}(x^a(k+1), w)$. Then, either the letter c remains unmatched in $W^{\text{FCFM}}(x^a(k), w)$, in which case $W^{\text{FCFM}}(x^a(k+1), w)$ and $W^{\text{FCFM}}(x^a(k), w)$ differ only at the index of that letter c in $W^{\text{FCFM}}(x^a(k), w)$, where there is a 0 in $W^{\text{FCFM}}(x^a(k+1), w)$; or c is matched with another letter b' in $W^{\text{FCFM}}(x^a(k), w)$, in which case we can repeat the same procedure for b' instead of b. As we have a finite number of letters in w, we eventually stop with a letter being present in a buffer and 0 in the other. In all cases, the buffers $W^{\text{FCFM}}(x^a(k+1), w)$ and $W^{\text{FCFM}}(x^a(k), w)$ differ only by one letter. □

For all $\tilde{x} \in \tilde{\mathbb{X}}$, let us define

$$T(\tilde{x}, a) = \text{Card}\left\{\text{letters } \tilde{x}_i \text{ of } \tilde{x} : \tilde{x}_i - a\right\}.$$

The above leads to the following result.

Corollary 1. *Let w be a word of length $2p$ such that for some $i \in [\![1;p]\!]$ and $j \in [\![p+1;p+i]\!]$, we have $w_i - w_j$. For such a pair $\{i, j\}$, and $k \in [\![0, 2p]\!]$, let*

$$x^{i,j}(k) = \begin{cases} x^{w_i}(k) & \text{if } k \in [\![0, p-1]\!], \\ \mathbf{0}_{2p} & \text{if } k = p, \\ x^{w_j}(2p-k) & \text{if } k \in [\![p+1, 2p]\!]. \end{cases}$$

Also let $u(k) = W^{\text{FCFM}}(x^{i,j}(k), w_1 \cdots w_p)$, for all $k \in [\![0, 2p]\!]$. Then there exists an integer k in $[\![0, 2p-1]\!]$ such that $u(k) = z_1 \cdots z_p$ differs from $u(k+1) = z'_1 \cdots z'_p$ by only one letter in some position l, where $z_l - w_j$, $z'_l = 0$, and for all $h \in [\![1, p]\!]$, $z'_h = w_h$ or $z'_h = 0$. Moreover we have that $T(u(k), w_j) = 1$ and $T(u(k+1), w_j) = 0$.

Proof. By Lemma 1, for all $k \in [\![0, 2p-1]\!]$, $u(k)$ and $u(k+1)$ differ at most by one letter in some position i (one being w_i, the other being a 0). Therefore, for all $k \in [\![0, 2p-1]\!]$, $|T(u(k), w_j) - T(u(k+1), w_j)| \leq 1$. Now notice that $2 \leq T(u(0), w_j)$, because gathering the words $x^{i,j}(0)$ and w would lead to at least $p+1$ w_i's out of $2p$ letters—so at least two w_i's must remain in $u(0)$. On the other hand, we also have that $T(u(2p), w_j) = 0$, because any letter of $w_1 \cdots w_p$ that can be matched with w_j gets matched in $u(2p)$ with the letters of $x^{i,j}(2p)$. As a consequence, there exists a rank $k \in [\![0, 2p-1]\!]$ such that $T(u(k), w_j) = 1$ and $T(u(k+1), w_j) = 0$. The remaining statements follow readily from Lemma 1. □

Theorem 4. *Consider a matching model with impatience (G, FCFM, μ, p). Let w be a word of length $2p$ of $\mathbb{V}^*$. Then the following conditions are equivalent:*

(i) w is a strongly synchronizing word;

(ii) w is a synchronizing word.

Proof. In view of Theorem 3, only the implication (ii) $\Rightarrow$ (i) remains to be proven. For this, we reason by contraposition. So let w be a word of length $2p$ such that $w_i - w_j$ for some $i \in [\![1;p]\!]$ and $j \in [\![p+1;p+i]\!]$. Let $i^* \in [\![1;p]\!]$ and $j^* \in [\![p+1;p+i^*]\!]$ be such that $w_{i^*} - w_{j^*}$ and $j^* = \inf\{j \in [\![p+1;2p]\!], \exists i \in [\![2p-j;p]\!]\ w_i - w_j\}$. We let $u = w_1 \cdots w_p$ and $v = w_{p+1} \cdots w_{2p}$. Let k^* be the integer obtained in Corollary 1 for $i \equiv i^*$ and $j \equiv j^*$. Then we get $u(k^*) = d_1 \cdots d_p$ and $u(k^*+1) = e_1 \cdots e_p$, where $u(.)$ is defined in Corollary 1. We will prove that $W^{\text{FCFM}}(u(k^*), v) \neq W^{\text{FCFM}}(u(k^*+1), v)$, which will show in turn that w is not a synchronizing word.

Let $i_1, \ldots, i_l$ be the indices (in increasing order) of letters of v that are matched with letters of $u(k^*)$ in $W^{\text{FCFM}}(u(k^*), v)$, and let $i'_1, \ldots, i'_h$ be the indices (in increasing order) of letters of v that are matched with letters of $u(k^*+1)$ in $W^{\text{FCFM}}(u(k^*+1), v)$. Now let us define the following sets:

$$\begin{cases} I_0 & = \emptyset, \\ I_{m+1} & = I_m \cup \left\{\inf\left\{j \in [\![p+1, p+m+1]\!] \setminus I_m : w_j - d_{m+1}\right\}\right\}, \quad m \in [\![0, p-1]\!]. \end{cases}$$

At each step of this construction we add to the set I_m the index of the letter that is matched with d_{m+1} in $W^{\text{FCFM}}(u(k^*), v)$, if any, as in FCFM, d_{m+1} is matched with the first compatible letter that has not been matched to a previous letter of u_k^*. In particular, we finally obtain that $I_p = \{i_1, \dots, i_l\}$. In the same way, we define the sets

$$\begin{cases} I'_0 &= \emptyset, \\ I'_{m+1} &= I'_m \cup \left\{\inf\left\{j \in [\![p+1, p+m+1]\!] \setminus I'_m : w_j - e_{m+1}\right\}\right\}, \quad m \in [\![0, p-1]\!], \end{cases}$$

and the same argument leads to $I'_p = \{i'_1, \dots, i'_h\}$.

If, in Corollary 1, the letter a that can be matched with w_{j^*} in $u(k^*)$ (and be replaced by a 0 in $u(k^*+1)$) is at position m, then by construction of j^*, $u(k^*)$, and $u(k^*+1)$, a will indeed be matched with w_{j^*} in $u(k^*)$. So the mth step is different for $W^{\text{FCFM}}(u(k^*), v)$ and $W^{\text{FCFM}}(u(k^*+1), v)$. For every other step m', as $d_{m'} \not- w_{j^*}$ and $e^{m'} \not- w_{j^*}$, we add the same letter, if any, to $I_{m'-1}$ and $I'_{m'-1}$. So we have $I_p = I'_p \cup \{j^*\}$. Let n_1 (resp., n_2) be the total number of letters from v that are matched in $W^{\text{FCFM}}(u(k^*), v)$ (resp., $W^{\text{FCFM}}(u(k^*+1), v)$). As the total numbers of matched letters are even, both $n_1 + |I_p|$ and $n_2 + |I'_p|$ are even. But as $|I_p|$ and $|I'_p|$ are of different parity, so are n_1 and n_2. Thus,

$$\begin{aligned} W^{\text{FCFM}}(x^{i^*,j^*}(k^*), w) = W^{\text{FCFM}}(u(k^*), v) &\neq W^{\text{FCFM}}(u(k^*+1), v) \\ &= W^{\text{FCFM}}(x^{i^*,j^*}(k^*+1), w), \end{aligned}$$

and w is not a synchronizing word. □

We have proven that being strongly synchronizing is a necessary and sufficient condition for being a synchronizing word of length $2p$ in the case where the matching policy is FCFM. This is not the case for all matching policies. For example, for the last-come-first-matched policy (lcfm), it can be proven that there exist synchronizing words of length $\lfloor \frac{3p}{2} \rfloor$, so that any suffix of those words that does not satisfy the p-condition would still be a synchronizing word. However, as we prove below, checking that a word is strongly synchronizing is a simple criterion that can be used to construct an efficient perfect sampling algorithm.

4.3.3. *A second perfect sampling algorithm.* We are now in position to introduce a perfect sampling algorithm for the state of a matching model with deterministic patience.

Definition 10. Consider a model (G, Φ, μ, p), and for all $k \in \mathbb{Z}$, define the SRS $\tilde{Y} := \left(\tilde{Y}_n^k\right)_{n \geq k}$ on the set $\tilde{\mathbb{Y}} = \{\emptyset\} \cup \bigcup_{j=1}^{2p} \mathbb{V}^j$ by

$$\begin{cases} \tilde{Y}_k^k &= \emptyset, \\ \tilde{Y}_{n+1}^k &= \tilde{g}(\tilde{Y}_n^k, V_{n+1}) \\ &:= \begin{cases} v_1 \cdots v_i V_{n+1}, & \text{if } \tilde{Y}_n^k = v_1 \cdots v_i \in \mathbb{V}^i \text{ with } i < 2p \\ v_2 \cdots v_{2p} V_{n+1}, & \text{if } \tilde{Y}_n^k = v_1 \cdots v_{2p} \in \mathbb{V}^{2p} \end{cases}, \quad n \geq k, \end{cases}$$

in such a way that for all k and all $n \geq k + 2p$, $\tilde{Y}_n^k$ represents the last $2p$ arrivals to the system at time n.

```
Data: A probability distribution $\mu$ on $\mathbb{V}$
$T_{down} \leftarrow -1$ ;
$T_{up} \leftarrow -2p$ ; /* We initialize the starting time at time $2p$ */
$\tilde{Y} \leftarrow \varnothing$ ;
while $\tilde{Y}$ is not strongly synchronizing do
    $i \leftarrow T_{up}$ ;
    $Y \leftarrow \varnothing$ ;
    for $j \leftarrow T_{up}$ to $T_{down}$ do
        draw $v_j$ from $\mu$; /* We draw the input at this iteration */
    end
    while $i < 0$ and $\tilde{Y}$ is not strongly synchronizing do
        $\tilde{Y} \longleftarrow g(\tilde{Y}, v_i)$; /* We investigate all arrival scenarios of length $2p$
         from $T_{up}$ to 0, and stop if one of them is strongly synchronizing.
         */
        $i \longleftarrow i+1$ ;
    end
    $T_{down} \leftarrow T_{up} - 1$ ;
    $T_{up} \leftarrow 2T_{up}$ ;
end
$\tilde{X} \longleftarrow z(\tilde{Y})$ ; /* We assign to $\tilde{X}$ the common state induced by the
 synchronizing word $\tilde{Y}$ */
/* We now transition $\tilde{X}$ to time $0$ as a matching system. */
while $i < 0$ do
    $\tilde{X} \longleftarrow \tilde{f}^{\Phi}(\tilde{X}, v_i)$ ;
    $i \longleftarrow i+1$ ;
end
return $\tilde{X}$
```

Algorithm 3: Simulation of the stationary probability of $\tilde{X}$ - Matching model with deterministic patience.

Consider Algorithm 3. It consists of another declination of Algorithm 1, started with $\tilde{Y} = \emptyset$, for $\tilde{Y}$ being the recursion of Definition 10, $b_1, \ldots, b_q$ the strongly synchronizing words of the model, and $a_1, \ldots, a_q$ the states of $\tilde{X}$ after the arrival of $b_1, \ldots, b_q$, respectively.

We have the following result.

Proposition 4. *$\tilde{X}$ is positive recurrent. Moreover, Algorithm 3 terminates almost surely, and its output is sampled from the stationary distribution of $\tilde{X}$.*

Proof. We can easily show that $\tilde{Y}$ r-controls $\tilde{X}$, with q the number of strongly synchronizing words. Let w be a strongly synchronizing word. By Theorem 3, w is a synchronizing word. Thus for all $k \in \mathbb{Z}$ and $n \geq k$, we get in particular that

$$\left[\tilde{Y}_n^k(\emptyset) = w\right] \Longrightarrow \left[\forall \tilde{x} \in \tilde{\mathbb{X}},\ \tilde{X}_n^k(\tilde{x}) = W^{\Phi}(\emptyset, w)\right], \tag{12}$$

which implies that $\tilde{Y}$ controls $\tilde{X}$ over all strongly synchronizing words. We conclude using Theorem 1. □

Remark 5. Observe that $\tilde{Y}$ is not irreducible; however, it reaches its recurrent class in $2p$ iterations. So for all strongly synchronizing words w, we still have that

$$\mathbb{P}(\tau_{\emptyset}^{\tilde{Y}}(w) < \infty) = 1.$$

4.3.4. *Efficiency of Algorithm 3.* In this section we analyze the coalescence time of Algorithm 3. For this, one needs to assess the probability that a given input word of length $2p$ is strongly synchronizing. This is, in turn, a function of μ and of the number of admissible arrival words of length $2p$ that are strongly synchronizing. The latter number is, clearly, highly dependent on the geometry of the compatibility graph at hand.

Let us first bound the average number of iterations of the algorithm to see the coalescence time, and then for the corresponding horizon in the past, as a function of the number of strongly synchronizing words. We have the following.

Proposition 5. *Let I be the number of iterations of Algorithm 1 to detect coalescence, and let $T = -p2^I$ be the corresponding starting time. Then we have that*

$$\mathbb{E}\left[-T\right] \leq \frac{2p}{\mathbb{P}^{p,\mu}},$$

where

$$\mathbb{P}^{p,\mu} = \mathbb{P}\left[V_1 \cdots V_{2p} \text{ is strongly synchronizing}\right].$$

Proof. For any integer $n \geq 1$, for all $i \in \mathbb{N}^*$, we let z_i^n be the word of length $2p$ representing the arrivals into the system between time $-p2^n + (i-1)2p$ and time $-p2^n + i2p - 1$, inclusive, in order of arrival. We also let

$$K^n = \inf\left\{i \in \mathbb{N}^* : z_i^n \text{ is strongly synchronizing}\right\}.$$

The independence of arrivals implies that the random variables K^n, $n \in \mathbb{N}^*$, are identically distributed (but not independent), with geometric distribution of parameter $\mathbb{P}^{p,\mu}$.

Now, it readily follows from Theorem 3 that for all $n \in \mathbb{N}^*$, $I \leq n$ in particular if there has been a strongly synchronizing arrival array between times $-p2^n$ and -1 inclusive, that is, if $2pK^n \leq p2^n$. Consequently, for all $n \in \mathbb{N}^*$ we get that

$$\mathbb{P}\left[-T > p2^n\right] = \mathbb{P}\left[I > n\right] \leq \mathbb{P}\left[2pK^n > p2^n\right] = \mathbb{P}\left[2pK^1 > p2^n\right].$$

This readily implies that $-T \leq_{\mathrm{st}} 2pK^1$, where $\leq_{\mathrm{st}}$ denotes the strong stochastic ordering. We deduce that

$$\mathbb{E}\left[-T\right] \leq 2p\mathbb{E}\left[K^1\right] = \frac{2p}{\mathbb{P}^{p,\mu}}.$$

□

Whenever the arrival measure μ is uniform over $\mathbb{V}$, the latter result specializes as follows.

Corollary 2. *If the graph $G = (\mathbb{V}, E)$ is of size n and μ is uniform over $\mathbb{V}$, we get the bounds*

$$\mathbb{E}\left[-T\right] \leq \frac{2pn^{2p}}{N(G,p)}, \qquad \mathbb{E}\left[I\right] \leq 1 + \frac{2pLogn - LogN(G,p)}{Log2},$$

where N(G,p) is the number of strongly synchronizing words of $\mathbb{V}^*$.

Proof. The results readily follow from Proposition 5, observing that in this case

$$\mathbb{P}^{p,\mu} = \frac{N(G,p)}{n^{2p}}.$$

□

For a given G and a given p, computing the number $N(G,p)$ of strongly synchronizing words is of crucial interest for assessing the efficiency of Algorithm 3. As Corollary 2 demonstrates, a function of the latter quantity provides bounds for the expected values of $|T|$ and I. We now turn to a specific evaluation of $N(G,p)$, and for this, we first need the following definitions.

Definition 11. Let $(G = (\mathbb{V}, E), \Phi, \mu, p)$ be a discrete matching model with impatience. For any strongly synchronizing word $w = w_1 \cdots w_{2p}$, the *trace* of w is defined as the word Z^w gathering, in order of their appearance, all distinct letters of the second half of w. In other words, we set

(1) $Z_1^w = w_{p+1}$;

(2) for all $i \in [\![1, p-1]\!]$,

$$Z_{i+1}^w = \begin{cases} Z_i^w & \text{if } w_{p+i+1} \in Z_i^w, \\ Z_i^w\, w_{p+i+1} & \text{if } w_{p+i+1} \notin Z_i^w, \end{cases}$$

and $Z^w \equiv Z_p^w$.

In what follows, for any word $z = z_1 \cdots z_l$, we denote by $\beta(z)$ the cardinality of the set of nodes that are incompatible with all letters of z; that is,

$$\beta(z) = \mathrm{Card}\left\{v \in \mathbb{V} : \forall i \in [\![1, l]\!],\, v \not- z_i\right\}.$$

We have the following.

Proposition 6. *Let $(G = (\mathbb{V}, E), \Phi, \mu, p)$ be a discrete matching model with impatience, and let $\mathscr{T}(G)$ be the set of words having distinct letters, that form a permutation of the elements of* *a set $U \subset \mathbb{V}$ such that $E(U) \neq \mathbb{V}$. Then the number N(G,p) of strongly synchronizing words is given by*

$$N(G,p) = \sum_{z=z_1\cdots z_l \in \mathscr{T}(G)} \sum_{\{1=k_1<k_2<\cdots<k_l<k_{l+1}=p+1\}} \prod_{i=1}^{l} i^{k_{i+1}-k_i-1} \beta(z_1 z_2 \cdots z_i)^{k_{i+1}-k_i}.$$

Proof. Let $z = z_1 \cdots z_l \in \mathbb{V}^*$ be a word having l distinct letters. For any word $w = w_1 \cdots w_{2p}$ of trace z, let us denote by

$$k_i^w = \inf\left\{ j \in [\![1, p]\!],\ w_{p+j} = z_i \right\}, \quad i \in [\![1, l]\!],$$

the consecutive indices, in the second half suffix of w, corresponding to the first occurrences of the successive letters of z.

Let $1 = k_1 < k_2 < \cdots < k_l < k_{l+1} = p+1$ be a fixed family of integers, and let w be a word of length $2p$. We first show the equivalence between the following two assertions:

(i) w is strongly synchronizing, has trace $z = z_1 \cdots z_l$, and satisfies

$$(k_1^w, k_2^w, \ldots, k_l^w) = (k_1, k_2, \ldots, k_l);$$

(ii) for all $i \in [\![1, l]\!]$,

(iia) $w_{p+k_i} = z_i$;
(iib) for all $j \in [\![k_i, k_{i+1}-1]\!]$, $w_{p+j} \in \{z_1, \cdots, z_i\}$ and $w_j \in E(\{z_1, \cdots, z_i\})^c$.

Indeed, if (i) holds true, then (ii) also holds by induction on i: first, (iia)–(iib) hold true for $i = 1$. Indeed, for all $j \in [\![k_1, k_2 - 1]\!]$ we have that $w_{p+j} = z_1$ by definition of k_1 and k_2, and thus, by definition of a strongly synchronizing word, that $w_j \not- w_{p+k_1} = z_1$. Now suppose that (iia)–(iib) hold true for some $i-1 \in [\![1, p-1]\!]$. Then (iia) holds for i by definition of the trace and of k_i. The statement (iib) also holds true by induction on j over $[\![k_i, k_{i+1}-1]\!]$: first, we have $w_{p+k_i} = z_i$ by (iia), implying, by definition of a strongly synchronizing word and in view of the induction assumption, that

$$\begin{aligned} w_{k_i} &\in E(\{w_\ell : \ell \in [\![p+1, p+k_i]\!]\})^c \\ &= E\left(\{w_\ell : \ell \in [\![p+1, p+k_i-1]\!]\} \cup \{w_{p+k_i}\}\right)^c \\ &= E(\{z_1, \cdots, z_i - 1\} \cup \{z_i\})^c = E(\{z_1, \cdots, z_i\})^c, \end{aligned}$$

so the properties in (iib) hold for $j = k_i$. Now suppose that they hold true for some $j-1 \in [\![k_i, k_{i+1}-2]\!]$. Then $w_{p+j} \in \{z_1, \cdots, z_i\}$ by the very definition of k_i. Thus, as w is strongly synchronizing, we have that

$$\begin{aligned} w_j &\in E(\{w_\ell : \ell \in [\![p+1, p+j]\!]\})^c \\ &= E\left(\{w_\ell : \ell \in [\![p+1, p+j-1]\!]\} \cup \{w_{p+j}\}\right)^c \\ &= E\left(\{z_1, \cdots, z_i\} \cup \{w_{p+j}\}\right)^c = E(\{z_1, \cdots, z_i\})^c. \end{aligned}$$

Thus (iib) holds true at index i, which completes the proof of (ii).

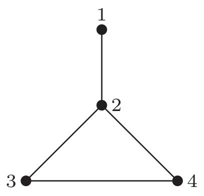

FIGURE 1. The paw graph.

Now suppose that (ii) holds. Then it follows from (iia) and the first property in (iib) that $k_i^w = k_i$ for all $i \in [\![1, l]\!]$. Now fix $j \in [\![1, p]\!]$, and let i be the index in $[\![1, l]\!]$ such that $j \in [\![k_i, k_{i+1} - 1]\!]$. Then, in view of (iia)–(iib), we get that

$$w_j \in E\left(\{z_1, \ \cdots \ , z_i\}\right)^c = E\left(\{w_{p+1}, \ \cdots \ , z_{p+j}\}\right)^c,$$

so w is indeed strongly synchronizing. From (ii), w also clearly has trace z, so (i) holds, which concludes the proof of (i) $\Leftrightarrow$ (ii).

Now, for a fixed trace z, to count the strongly synchronizing words having trace z, it thus suffices to count, for all families of integers $1 = k_1 < k_2 < \cdots < k_l < k_{l+1} = p + 1$, all the words w satisfying (ii). First, the letters at indices $k_1, \ldots, k_l$ are fixed, and for all $i \in [\![1, l]\!]$ we have i possibilities for each letter between indices $k_i + 1$ and $k_{i+1} - 1$, and $\beta(z_1 \cdots z_i)$ possibilities for each letter between indices $k_i - p$ and $k_{i+1} - 1 - p$. Therefore, the number of strongly synchronizing words having trace z is given by

$$N_z := \sum_{\{1=k_1<k_2<\cdots<k_l<k_{l+1}=p+1\}} \prod_{i=1}^{l} i^{k_{i+1}-k_i-1} \beta(z_1 z_2 \cdots z_i)^{k_{i+1}-k_i}. \tag{13}$$

Lastly, to get $N(G, p)$ we must sum the above quantity over all possible traces of strongly synchronizing words. To characterize this set, observe that any trace z necessarily has distinct letters, forming a permutation of a set $U_z \subset \mathbb{V}$. If $E(U_z) \neq \mathbb{V}$, then there exists a letter $i \in \mathbb{V} \setminus E(U_z)$, and z clearly is the trace of any word w whose prefix of size p is $ii \ \cdots \ i$, and whose suffix of size p is a permutation of the elements of U. Now, if $E(U_z) = \mathbb{V}$, for any strongly synchronizing word w having trace z we must have that $w_p \notin E(U_z)$, leading to an immediate contradiction. Thus, for z to be a trace, it is necessary and sufficient that $E(U_z) \neq \mathbb{V}$, which concludes the proof. □

4.3.5. *Example.* To illustrate the efficiency of Algorithm 3 in the case of deterministic patience times, we consider the simple non-trivial example of the so-called paw graph G of Figure 1.

As the above results demonstrate, for any p, to compute the number $N(G, p)$ of strongly synchronizing words, we first need to determine the set of all possible traces $\mathscr{T}(G)$ of G. In the present case we readily obtain that

$$\mathscr{T}(G) = \{1, 2, 3, 4, 13, 14, 31, 34, 41, 43, 134, 143, 314, 341, 413, 431\}.$$

Indeed, any trace containing a 2 can only contain 2, since adding another class will result in having compatible classes, which cannot be the case for a trace. Conversely, any other word containing only 1 or 3 or 4 is possible as a trace, since not having 2 in the word means that the letters of the word are still compatible with the class 1. It is then immediate to compute $\beta(z)$ for all $z \in \mathscr{T}(G)$ using (13). We obtain

$$\begin{cases} N_1 = 3^p,\ N_2 = 1,\ N_3 = 2^p,\ N_4 = 2^p, N_{13} = \frac{3}{2}4^p - 2.3^p,\ N_{14} = \frac{3}{2}4^p - 2.3^p, \\ N_{31} = \frac{1}{2}4^p - 2^p,\ N_{34} = (p-1)2^{p-1},\ N_{41} = \frac{1}{2}4^p - 2^p,\ N_{43} = (p-1)2^{p-1}, \\ N_{134} = 3.2^{2p-1} - (2p+4)3^{p-1},\ N_{143} = 3.2^{2p-1} - (2p+4)3^{p-1}, \\ N_{314} = 2^{2p-1} - 4.3^{p-1} + 2^p,\ N_{341} = 2.3^{p-1} - (p+1)2^{p-1}, \\ N_{413} = 2^{2p-1} - 4.3^{p-1} + 2^p,\ N_{431} = 2.3^{p-1} - (p+1)2^{p-1}. \end{cases} \tag{14}$$

For clarity, let us detail one of the above computations, for $z = 13$. We then have $\beta(1) = |\{1, 3, 4\}| = 3$ and $\beta(13) = |\{1, 3\}| = 2$. Therefore, using (13) we have

$$\begin{aligned} N_{13} &= \sum_{k_2=2}^{p} 1^{k_2-1-1}\beta(1)^{k_2-1}2^{p+1-k_2-1}\beta(13)^{p+1-k_2} \\ &= \sum_{k_2=2}^{p} 3^{k_2-1}2^{p-k_2}2^{p+1-k_2} \\ &= \frac{2\times 4^p}{3}\sum_{k_2=2}^{p}\left(\frac{3}{4}\right)^{k_2} = \frac{3}{2}4^p - 2.3^p. \end{aligned}$$

Summing all elements of (14) and rearranging, we obtain that

$$N(G, p) = 1 + 2^{2p+3} - 3^{p+1} - 4(p+3)3^{p-1}.$$

Then, applying Corollary 2 and Jensen's inequality, for the average number of iterations needed by Algorithm 1 to detect coalescence, we obtain the bound

$$\begin{aligned} \mathbb{E}[I] &\le 1 + \frac{2p\,\mathrm{Log}n - \mathrm{Log}N(G, p)}{\mathrm{Log}2} \\ &= 1 + 4p - \frac{\mathrm{Log}\left(1 + 2^{2p+3} - 3^{p+1} - 4(p+3)3^{p-1}\right)}{\mathrm{Log}2} =: B_I, \end{aligned}$$

and the average starting time T to detect coalescence is bounded by $-p2^{B_I}$. In Table 1, we specify the number of strongly synchronizing words, together with the corresponding bounds for $\mathbb{E}[I]$ and $\mathbb{E}[-T]$, for various values of p.

4.3.6. *Complexity comparison.* Having provided a bound for the average coalescence time for Algorithm 3, we now compare the number of operations necessary to complete Algorithm 3 to the number of operations necessary to complete the primitive CFTP algorithm, which consists of running chains started from all possible states in parallel. To compare these two algorithms, we need to specify what we mean by *operations*: we say that an algorithm does one operation if it compares two letters of $\mathbb{V}$ to determine whether they are equal or not, or whether the two letters are connected in G. It is intuitively clear that each of the two algorithms can basically be decomposed into a sequence of such operations:

TABLE 1. Efficiency of Algorithm 1.

p	$N(G, p)$	Bound for $\mathbb{E}[I]$	Bound for $\mathbb{E}[-T]$
1	8	2	4
2	42	3,608	24,381
3	216	5,245	113,778
4	1050	6,964	499,322
5	4872	8,750	2152,250
6	21834	10,586	9220,784
7	95352	12,460	39412,874
8	408378	14,360	168274,189
9	1723176	16,283	717831,830
10	7187946	18,223	3059320,779

- In the CFTP algorithm, the matching of the incoming individuals amounts to an investigation of the set of stored compatible items in a determinate order, and thus to a sequence of such operations. Second, so does the test of equality of the current states of all Markov chains, at any given time.
- In Algorithm 3, testing the 'strongly synchronizing' property at all times is again a sequence of operations, and so is the construction of the dynamics of the recursion, from the coalescence time onward.

To estimate the number of operations in the two algorithms, for two values of p (3 and 6), we first drew realizations of Erdös–Rényi graphs G of parameters (n, α), which are conditioned to be connected, for various values of the size n and of the connectivity parameter α. We then tracked the average number of operations for 10 realizations of both algorithms, on the same graph each time.

The results, presented in Tables 2–5, tend to indicate that Algorithm 3 is much more efficient than primitive CFTP, and that the performance gap is particular important for sparse graphs. This last fact is an intuitively clear consequence of the fact that the proportion of strongly synchronizing words is decreasing in the number of edges. For $\alpha \geq \frac{3}{4}$, however, we observe cases where Algorithm 3 does not terminate in a reasonable amount of time.

4.4. Deterministic matching model with latency

It is well known that the primitive CFTP algorithm is in general not a good benchmark in terms of complexity, as it requires the coalescence of a large number of versions of the Markov chain considered—which makes its use impractical for a large state space. On the other hand, as previously mentioned, non-trivial deterministic matching models do not satisfy Condition (9). This means that the SRS defined by the recursion (2) cannot hit 0, and so we cannot use Algorithm 2 for deterministic matching models.

In this section we introduce the following variant of the model of Section 4.3: we suppose that latency is allowed; that is, at each instant we suppose that, with a positive probability γ, no item enters the system. In other words, the generic inter-arrival time ξ follows a geometric law of parameter $1 - \gamma$. The sequences $\left(\widehat{V}_n\right)_{n\in\mathbb{Z}}$ and $\left(\widehat{P}_n\right)_{n\in\mathbb{Z}}$ are then defined as follows:

TABLE 2. Average number of operations of the algorithms for 10 repetitions with $p = 3$ and multiple values of (n, α).

$p = 3$	$\alpha = \frac{1}{8}$	$\alpha = \frac{2}{8}$	$\alpha = \frac{3}{8}$	$\alpha = \frac{4}{8}$	$\alpha = \frac{5}{8}$
$n = 4$, CFTP	2907.67	3356.04	3012.46	3228.72	3393.52
$n = 4$, Algo 3	123.16	168.11	297.64	213.27	307.5
$n = 5$, CFTP	4689.57	5078.86	4694.66	5542.87	4713.41
$n = 5$, Algo 3	102.2	108.24	161.58	422.99	548.35
$n = 6$, CFTP	7888.29	7350.42	6550.72	7466.3	7319.94
$n = 6$, Algo 3	93.96	82.39	161.96	338.36	406.9
$n = 7$, CFTP	11458.40	10200.46	111044.26	10455.48	9222.91
$n = 7$, Algo 3	69.06	117.94	140.82	252.71	764.8
$n = 8$, CFTP	15984.74	14829.7	15127.1	12565.42	12189.06
$n = 8$, Algo 3	56.85	86.28	93.19	241.3	818.16

TABLE 3. Average CPU time of the algorithms on a standard computer for 10 repetitions with $p = 3$ and multiple values of (n, α).

$p = 3$	$\alpha = \frac{1}{8}$	$\alpha = \frac{2}{8}$	$\alpha = \frac{3}{8}$	$\alpha = \frac{4}{8}$	$\alpha = \frac{5}{8}$
$n = 4$, CFTP	0.00969	0.01	0.00890	0.00937	0.01
$n = 4$, Algo 3	0.00078	0.00093	0.00172	0.00125	0.00172
$n = 5$, CFTP	0.01282	0.01468	0.01328	0.01609	0.01422
$n = 5$, Algo 3	0.00047	0.00047	0.00125	0.00280	0.00328
$n = 6$, CFTP	0.02236	0.02046	0.01875	0.02188	0.02375
$n = 6$, Algo 3	0.00063	0.00062	0.00110	0.00220	0.00233
$n = 7$, CFTP	0.03499	0.03608	0.03281	0.03407	0.02954
$n = 7$, Algo 3	0.00046	0.00062	0.00079	0.00187	0.00484
$n = 8$, CFTP	0.04984	0.04545	0.04404	0.03125	0.02922
$n = 8$, Algo 3	0.00048	0.00078	0.00078	0.00172	0.00594

- If a n item enters the system at time n, then $\widehat{V}_n$ is the class of the item entering the system, and $\widehat{P}_n = p + \varepsilon$ is the patience of the item entering the system.
- Otherwise, we set $\widehat{V}_n = -1$ and $\widehat{P}_n = 0$.

We denote such a deterministic model with latency by $(G, \Phi, \mu, p, \gamma)$. Similarly as in Section 4.3, we then easily obtain a simplified representation of the system state.

Definition 12. For all $n \in \mathbb{Z}$, the *word-profile* of the system just before time n is defined by the word

$$\widehat{X}_n = w_1 \cdots w_p \in (\mathbb{V} \cup \{0\} \cup \{-1\})^p,$$

TABLE 4. Average number of operations of the algorithms for 10 repetitions with $p = 6$ and multiple values of (n, α).

$p = 6$	$\alpha = \frac{1}{8}$	$\alpha = \frac{2}{8}$	$\alpha = \frac{3}{8}$	$\alpha = \frac{4}{8}$	$\alpha = \frac{5}{8}$
$n = 4$, CFTP	467834.38	477424.56	457725.73	433542.6	363723.37
$n = 4$, Algo 3	5576.1	12843.9	12070.8	21559.18	17672.3
$n = 5$, CFTP	1248551.28	1139776.29	919830.29	853625.28	980490.29
$n = 5$, Algo 3	12472.92	11666.23	11111.01	58257.99	143122.55
$n = 6$, CFTP	3218753.63	2446069.32	2999130.02	2128500.86	1547150.53
$n = 6$, Algo 3	3183.32	6274.56	11272.88	127429.52	284116.14
$n = 7$, CFTP	4790047.9	7288225.3	7117622.9	2536934.7	3628303.2
$n = 7$, Algo 3	2609.97	9818.73	455.36	171196.79	381580.97
$n = 8$, CFTP	14779764.95	10594880.96	8477686.16	6073463.72	4123539.06
$n = 8$, Algo 3	2382.32	2174.99	14180.58	46050.05	389028.98

TABLE 5. Average CPU time of the algorithms on a standard computer for 10 repetitions with $p = 6$ and multiple values of (n, α).

$p = 3$	$\alpha = \frac{1}{8}$	$\alpha = \frac{2}{8}$	$\alpha = \frac{3}{8}$	$\alpha = \frac{4}{8}$	$\alpha = \frac{5}{8}$
$n = 4$, CFTP	1.10219	1.20124	1.13655	1.09686	0.91844
$n = 4$, Algo 3	0.05930	0.1313	0.1328	0.24062	0.2062
$n = 5$, CFTP	2.85954	2.66343	2.06705	2.23469	2.66578
$n = 5$, Algo 3	0.06749	0.08233	0.06798	0.55468	2.4180
$n = 6$, CFTP	7.99048	6.53797	9.8092	5.83189	4.18533
$n = 6$, Algo 3	0.01812	0.03922	0.07391	1.31535	3.8147
$n = 7$, CFTP	17.61454	14.21235	15.81282	10.9244	17.6595
$n = 7$, Algo 3	0.01516	0.06514	0.03921	3.53732	7.23748
$n = 8$, CFTP	42.85513	35.96576	27.95357	18.95672	12.183
$n = 8$, Algo 3	0.01517	0.01281	0.09954	46.98482	13.43019

where for all $i \in [\![1, p]\!]$,

$$w_i = \begin{cases} \widehat{V}_{n-p+i-1} = v \in \mathbb{V} & \text{if the item of class } v \text{ entering at time } n-p+i-1 \\ & \text{ was not matched before } n, \\ 0 & \text{if the item entering at time } n-p+i-1 \\ & \text{ was matched before } n, \\ \widehat{V}_{n-p+i-1} = -1 & \text{if no item entered at time } n-p+i-1. \end{cases} \quad .$$

We can therefore view the latency at a certain time n as the arrival of an item labeled -1 that cannot be matched with any other item. We then denote by

$$\widehat{\mathbb{X}} = (\mathbb{V} \cup \{0\} \cup \{-1\})^p$$

the (finite) state space space of $\widehat{X}$. In contrast to the model of Section 4.3 (which can be seen as a particular of the present one for $\gamma = 0$), (9) is satisfied here, since the geometric random variable ξ can be arbitrarily large. Therefore, Algorithm 2 can be used to design a perfect sampling algorithm in this case. Let $\widehat{Y} := \left(\widehat{Y}_n\right)_{n\in\mathbb{Z}}$ be the SRS defined by the recursive equation

$$\widehat{Y}_{n+1} = \left[\max\left(\widehat{Y}_n, \widehat{P}_n\right) - 1\right]^+ = g\left(\widehat{Y}_n, \widehat{P}_n\right), \quad n \in \mathbb{Z}. \tag{15}$$

Proposition 7. *For any $n \in \mathbb{Z}$, the following statements are equivalent:*

(i) $\widehat{Y}_n = 0$.

(ii) For all $k \in [\![1, p]\!]$, $\widehat{V}_{n-k} = -1$ (and equivalently $\widehat{P}_{n-k} = 0$).

Proof. Fix $n \in \mathbb{Z}$. Regarding the implication (ii) $\Rightarrow$ (i), by the construction of $\widehat{P}$ and $\widehat{Y}$ we have that

$$\widehat{Y}_{n-p} \le p + \varepsilon - 1.$$

Moreover, for all $k \in [\![1, p]\!]$,

$$\widehat{Y}_{n-k+1} = \left[\max\left(\widehat{Y}_{n-k}, \widehat{P}_{n-k}\right) - 1\right]^+ = \left[\max\left(\widehat{Y}_{n-k}, 0\right) - 1\right]^+ = \left[\widehat{Y}_{n-k} - 1\right]^+,$$

so by induction we obtain that

$$\widehat{Y}_n \le \max\left(\widehat{Y}_{n-p} - p, 0\right) \le \max\left(p+\varepsilon - 1 - p, 0\right) = 0.$$

We now turn to the converse implication (i) $\Rightarrow$ (ii). Suppose to the contrary that for some $k \in [\![1, p]\!]$ we have $\widehat{P}_{n-k} \neq 0$, which, by the very definition of $\widehat{P}$, means that $\widehat{P}_{n-k} = p + \varepsilon$. Then, as $\widehat{Y}_{n-k} \le p + \varepsilon - 1$, we have that

$$\widehat{Y}_{n-k+1} = \left[\max\left(\widehat{Y}_{n-k}, \widehat{P}_{n-k}\right) - 1\right]^+ = \widehat{P}_{n-k} - 1 = p + \varepsilon - 1.$$

But for all $l \in [\![1, k-1]\!]$ we have that $\widehat{Y}_{n-l+1} \ge \widehat{Y}_{n-l} - 1$, so that by an immediate induction,

$$\widehat{Y}_n \ge p + \varepsilon - 1 - (k-1) \ge \varepsilon > 0.$$

□

As a consequence of Proposition 7, determining when $\widehat{Y} = 0$ in Algorithm 2 amounts to checking that the last p arrivals are all -1's, meaning that no item has entered the system in the last p instants. In fact, as (ii) above has a positive probability, we immediately see that in the present context, Algorithm 2 terminates almost surely.

For any $x = x_1 \cdots x_p \in \widehat{\mathbb{X}}$, denote by $\overset{\circ}{x}$ the word $\overset{\circ}{x}_1 \cdots \overset{\circ}{x}_p$, where for all $i \in [\![1, p]\!]$, $\overset{\circ}{x}_i = x_i \mathbf{1}_{x_i \neq -1}$. The notions of synchronizing and strongly synchronizing words are then extended as follows.

Definition 13. A word $w \in (\mathbb{V} \cup -1)^*$ is said to be *synchronizing* for the deterministic matching model with latency $(G, \Phi, \mu, p, \gamma)$ if

$$\exists z \in \widehat{\mathbb{X}},\ \forall x \in \widehat{\mathbb{X}},\ W^{\Phi}(x, w) = w_x \text{ is such that } \overset{\circ}{w}_x = z. \tag{16}$$

We say that a word $w = w_1 \cdots w_{2p} \in (\mathbb{V} \cup -1)^*$ is *strongly synchronizing* if

$$\forall i \in [\![1, p]\!],\ \forall j \in [\![p+1, p+i]\!],\ w_i \not\!\!- w_j,$$

where, by convention, for all $v \in \mathbb{V}$, $v \not\!\!- -1$.

TABLE 6. Average CPU time of the algorithms on a standard computer for 100 repetitions with $p = 3$, $\gamma = 0.2$, and multiple values of (n, α).

$p = 3$	$\alpha = \frac{1}{8}$	$\alpha = \frac{2}{8}$	$\alpha = \frac{3}{8}$	$\alpha = \frac{4}{8}$	$\alpha = \frac{5}{8}$
$n = 4$, Algo 2	0.0038045	0.0037805	0.0040165	0.0039802	0.0042959
$n = 4$, Algo 3	0.0004670	0.0004578	0.0005623	0.0007405	0.0009837
$n = 5$, Algo 2	0.0044388	0.0041361	0.0043400	0.0045547	0.0043607
$n = 5$, Algo 3	0.0004592	0.0005210	0.0005462	0.0006469	0.0008396
$n = 6$, Algo 2	0.0046643	0.0045620	0.0046185	0.0046222	0.0048220
$n = 6$, Algo 3	0.0004305	0.0004332	0.0005398	0.0006290	0.0010127
$n = 7$, Algo 2	0.0049668	0.0053617	0.0051719	0.0048706	0.0047918
$n = 7$, Algo 3	0.0003674	0.0005255	0.0006371	0.0008323	0.0009967
$n = 8$, Algo 2	0.0049822	0.0049859	0.0050439	0.0052366	0.0059534
$n = 8$, Algo 3	0.0003490	0.0003939	0.0006275	0.0008972	0.0013853

We can then apply the exact same arguments as for Theorems 3 and 4 to show the following.

Proposition 8. *Any strongly synchronizing word $w \in (\mathbb{V} \cup -1)^*$ is also a synchronizing word for the deterministic matching model with latency $(G, \Phi, \mu, p, \gamma)$. Conversely, if the matching policy Φ is* FCFM*, then any synchronizing word of length $2p$ is strongly synchronizing.*

The previous result implies that Algorithm 3 (by taking strongly synchronizing words in this new sense) terminates almost surely, and also produces a sample of the stationary distribution of the model with latency.

In conclusion, for a model with latency, both Algorithm 2 and Algorithm 3 are valid perfect sampling algorithms that terminate almost surely, and we can now compare their performance. For this, first notice that Algorithm 3 is obviously faster than Algorithm 2, since the arrival of p consecutive -1's also creates a strongly synchronizing word, as any word of length $2p$ in which the first or last p letters are all -1 is strongly synchronizing. In Tables 6–9, we quantify the gain from applying Algorithm 3 rather than Algorithm 2 in terms of CPU time, for various parameters.

4.5. Estimating of the loss probability for ML and FCFM

Algorithm 3 returns a random variable that is distributed from the stationary distribution of the system. This result can be of critical use in comparing the performance of systems for which no exact characterization of the steady state is known. As an example, we are able to assess the asymptotic loss rate of items of every class. We use this to compare two matching policies in steady state: match-the-longest (ML) and first-come-first-matched (FCFM).

Let $(G = (\mathbb{V}, E), \Phi, \mu, p)$ be a discrete matching model with deterministic impati ence, and let $\tilde{X} = \left(\tilde{X}_n\right)_{n\in\mathbb{Z}}$ be the Markov chain of the system. Let π be the stationary distribution for X, and for all $(i, j) \in \mathbb{V}^2$ such that $(i, j) \notin E$, let

$$A_{i,j} = \{x = x_1 \cdots x_j \in \mathbb{X},\ x_1 = i \text{ and the arrival is of class } j \text{ in a buffer } x\}.$$

TABLE 7. Average CPU time of the algorithms on a standard computer for 10^4 repetitions with $p = 3$, $\gamma = 0.5$, and multiple values of (n, α).

$p = 3$	$\alpha = \frac{1}{8}$	$\alpha = \frac{2}{8}$	$\alpha = \frac{3}{8}$	$\alpha = \frac{4}{8}$	$\alpha = \frac{5}{8}$
$n = 4$, Algo 2	0.0003215	0.0003227	0.0003087	0.0003241	0.0002954
$n = 4$, Algo 3	0.0002715	0.0002666	0.0002210	0.0002102	0.0002593
$n = 5$, Algo 2	0.0003761	0.0003799	0.0003718	0.0003690	0.0003190
$n = 5$, Algo 3	0.0002000	0.0001950	0.0001834	0.0002102	0.0002446
$n = 6$, Algo 2	0.0004253	0.0003487	0.0003704	0.0003690	0.0003768
$n = 6$, Algo 3	00.0002045	0.0001953	0.0002090	0.0002409	0.0002559
$n = 7$, Algo 2	0.0004097	0.0003916	0.0004289	0.0004132	0.0004103
$n = 7$, Algo 3	0.0002384	0.0002094	0.0002073	0.0002868	0.0002901
$n = 8$, Algo 2	0.0004186	0.0004070	0.0004908	0.0005598	0.0006104
$n = 8$, Algo 3	0.0002074	0.0001791	0.0002156	0.0002520	0.0002659

TABLE 8. Average CPU time of the algorithms for 100 repetitions with $p = 6$, $\gamma = 0.2$, and multiple values of (n, α).

$p = 6$	$\alpha = \frac{1}{8}$	$\alpha = \frac{2}{8}$	$\alpha = \frac{3}{8}$	$\alpha = \frac{4}{8}$	$\alpha = \frac{5}{8}$
$n = 4$, Algo 2	0.7051800	0.5984500	0.5163900	0.4517600	0.7003300
$n = 4$, Algo 3	0.0034300	0.0057900	0.0056100	0.0068200	0.0246900
$n = 5$, Algo 2	0.5984500	0.8970700	0.5821500	0.6060300	0.5139100
$n = 5$, Algo 3	0.0035900	0.0026300	0.0065800	0.0109600	0.0221800
$n = 6$, Algo 2	0.7514100	0.6232700	0.6096600	0.9526300	0.7364400
$n = 6$, Algo 3	0.0009300	0.0042500	0.0051800	0.0105000	0.0219500
$n = 7$, Algo 2	0.7517200	0.5460800	0.4714300	0.5977900	0.6670700
$n = 7$, Algo 3	0.0014000	0.0025000	0.0042300	0.0084900	0.0171500
$n = 8$, Algo 2	0.5546900	0.6359300	0.4876600	0.7620300	0.6122400
$n = 8$, Algo 3	0.00109000	0.0017500	0.0037400	0.0092300	0.0376300

TABLE 9. Average CPU time of the algorithms on a standard computer for 10^4 repetitions with $p = 6$, $\gamma = 0.5$, and multiple values of (n, α).

$p = 6$	$\alpha = \frac{1}{8}$	$\alpha = \frac{2}{8}$	$\alpha = \frac{3}{8}$	$\alpha = \frac{4}{8}$	$\alpha = \frac{5}{8}$
$n = 4$, Algo 2	0.0046900	0.00316360	0.0034701	0.0031292	0.0031772
$n = 4$, Algo 3	0.0010425	0.0010735	0.0012352	0.0013639	0.0016111
$n = 5$, Algo 2	0.0040600	0.0035472	0.0035068	, 0.0034339	0.0035351
$n = 5$, Algo 3	0.0008387	0.0010067	0.0010954	0.0013510	0.0015984
$n = 6$, Algo 2	0.0034200	0.0035355	0.0039241	0.00358213	0.0036049
$n = 6$, Algo 3	0.0008432	0.0008729	0.0010616	0.0013186	0.0017617
$n = 7$, Algo 2	0.0048400	0.0037248	0.0037925	0.0036611	0.0038020
$n = 7$, Algo 3	0.0007764	0.0008591	0.0010278	0.0013062	0.0017764
$n = 8$, Algo 2	0.0054600	0.0038513	0.0038263	0.0037788	0.0038358
$n = 8$, Algo 3	0.0007209	0.0008657	0.0010599	0.0013521	0.0018399

TABLE 10. Monte Carlo estimates for the asymptotic loss rates for 10^4 repetitions of Algorithm 3 for a random Erdös–Rényi graph of parameters $n = 5$, $\alpha = 0.6$, for $p = 5$ and μ the uniform distribution.

	ρ	$\rho(1)$	$\rho(2)$	$\rho(3)$	$\rho(4)$	$\rho(5)$
FCFM	0.0293	0.00026	0.00032	0.0132	0.0152	0.00032
ML	0.03122	0.00028	0.0003	0.01536	0.01486	0.00042

The asymptotic loss rate of items of class i is denoted by

$$\rho(i) := \lim_{N \to +\infty} \frac{\sum_{n=1}^{N} \mathbb{1}_{A_n^i}}{N}, \tag{17}$$

where for all n,

$$A_n^i = \{\text{an item of class } i \text{ is lost at time } n\}.$$

An immediate first-step analysis implies that

$$\rho(i) = \sum_{j \in \mathbb{V}} \pi(A_{i,j})\mu(j), \tag{18}$$

so $\rho(i)$ can also be interpreted as the probability of losing an item of class i in the system at a given instant, in steady state. Reasoning similarly, we have that

$$\rho = \sum_{i \in \mathbb{V}} \rho(i)$$

is the asymptotic loss rate of items (of any class) in the system, and can also be seen as the probability of losing an item (of any class) at a given time, in steady state. Using Equation 18, we can then estimate these asymptotic loss rates by running our perfect simulation Algorithm 3, and then estimating $\pi(A_{i,j})$ for all $i, j \in \mathbb{V}$ using a Monte Carlo estimate.

Table 10 presents the results over 10^4 simulations, for G a random Erdös–Rényi graph of parameters $n = 5$, $\alpha = 0.6$, conditioned on being connected, for $p = 5$, and for μ the uniform distribution. Both matching policies FCFM and ML are implemented on the same samples each time. We observe that the overall asymptotic loss rate is slightly, but consistently, lower under FCFM than under ML, although nominal loss rates of given nodes can be higher under FCFM.

Funding information

This research was funded by the research grant ANR-18-CE40-0019-01, 'MATCHES', and by a Région Grand Est PhD grant.

Competing interests

There were no competing interests to declare which arose during the preparation or publication process of this article.

References

[1] Adan, I., Bušić, A., Mairesse, J. and Weiss, G. (2018). Reversibility and further properties of FCFS infinite bipartite matching. *Math. Operat. Res.* **43**, 598–621.

[2] Adan, I., Kleiner, I., Righter, R. and Weiss, G. (2018). FCFS parallel service systems and matching models. *Performance Evaluation* **127**, 253–272.

[3] Adan, I. and Weiss, G. (2012). Exact FCFS matching rates for two infinite multitype sequences. *Operat. Res.* **60**, 475–489.

[4] Anantharam, V. and Konstantopoulos, T. (1999). A correction and some additional remarks on 'Stationary solutions of stochastic recursions describing discrete event systems'. *Stoch. Process. Appl.* **80**, 271–278.

[5] Aveklouris, A., DeValve, L., Ward, A. R. and Wu, X. (2021). Matching impatient and heterogeneous demand and supply. Preprint. Available at https://arxiv.org/abs/2102.02710.

[6] Baccelli, F. and Brémaud, P. (2003). *Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences*. Springer, Berlin, Heidelberg.

[7] Begeot, J., Marcovici, I. and Moyal, P. (2023). Stability regions of systems with compatibilities, and ubiquitous measures on graph. *Queueing Systems* **103**, 275–312.

[8] Begeot, J., Marcovici, I., Moyal, P. and Rahme, Y. (2021). A general stochastic matching model on multigraphs. *ALEA* **18**, 1325–1351.

[9] Blanchet, J. and Dong, J. (2015). Perfect sampling for infinite server and loss systems. *Adv. Appl. Prob.* **47**, 761–786.

[10] Blanchet, J., Dong, J. and Pei, Y. (2018). Perfect sampling of GI/GI/c queues. *Queueing Systems* **90**, 1–33.

[11] Borovkov, A. and Foss, S. (1992). Stochastically recursive sequences and their generalizations. *Siberian Adv. Math.* **2**, 16–92.

[12] Borovkov, A. and Foss, S. (1994). Two ergodicity criteria for stochastically recursive sequences. *Acta Appl. Math.* **34**, 125–134.

[13] Borovkov, A. A. (1998). *Ergodicity and Stability of Stochastic Processes*. John Wiley, New York.

[14] Bušić, A., Gaujal, B. and Vincent, J.-M. (2008). Perfect simulation and non-monotone Markovian systems. In *ValueTools '08: Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools*, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Brussels.

[15] Bušić, A., Gupta, V. and Mairesse, J. (2013). Stability of the bipartite matching model. *Adv. Appl. Prob.* 45, 351–378.

[16] Caldentey, R., Kaplan, E. and Weiss, G. (2009). FCFS infinite bipartite matching of servers and customers. *Adv. Appl. Prob.* **41**, 695–730.

[17] Castro, F., Nazerzadeh, H. and Yan, C. (2020). Matching queues with reneging: a product form solution. *Queueing Systems* **96**, 359–385.

[18] Comte, C. (2022). Stochastic non-bipartite matching models and order-independent loss queues. *Stoch. Models* **38**, 1–36.

[19] Comte, C., Mathieu, F. and Bušić, A. (2021). Stochastic dynamic matching: a mixed graph-theory and linear-algebra approach. Preprint. Available at https://arxiv.org/abs/2112.14457.

[20] Connor, S. B. and Kendall, W. S. (2007). Perfect simulation for a class of positive recurrent Markov chains. *Ann. Appl. Prob.* **17**, 781–808.

[21] Decreusefond, L. and Moyal, P. (2012). *Stochastic Modeling and Analysis of Telecom Networks*. John Wiley, Hoboken, NJ.

[22] Foss, S. G. and Tweedie, R. L. (1998). Perfect simulation and backward coupling. *Stoch. Models* **14**, 187–203.

[23] Green, P. and Murdoch, D. (1999). Exact sampling for Bayesian inference: towards general purpose algorithms (with discussion). In *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting, June 6–10, 1998*, Oxford University Press, pp. 301–321.

[24] Huber, M. (2004). Perfect sampling using bounding chains. *Ann. Appl. Prob.* **14**, 734–753.

[25] Huber, M. L. (2016). *Perfect Simulation*. CRC Press, Boca Raton, FL.

[26] Jonckheere, M., Moyal, P., Ramrez, C. and Soprano-Loto, N. (2023). Generalized max-weight policies in stochastic matching. *Stoch. Systems* **13**, 40–58.

[27] Kelly, F. P. (1991). Loss networks. *Ann. Appl. Prob.* **1**, 319–378.

[28] Kendall, W. (2004). Geometric ergodicity and perfect simulation. *Electron. Commun. Prob.* **9**, 140–151.

[29] Kendall, W. S. (1998). Perfect simulation for the area-interaction point process. In *Probability Towards 2000*, Springer, New York, pp. 218–234.

[30] Kendall, W. S. and Moller, J. (2000). Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Adv. Appl. Prob.* **32**, 844–865.

[31] LISEK, B. (1982). A method for solving a class of recursive stochastic equations. *Z. Wahrscheinlichkeitsth.* **60**, 151–161.
[32] MAIRESSE, J. AND MOYAL, P. (2016). Stability of the stochastic matching model. *J. Appl. Prob.* **53**, 1064–1077.
[33] MEYN, S. P. AND TWEEDIE, R. L. (2012). *Markov Chains and Stochastic Stability*. Springer, New York.
[34] MOYAL, P. (2008). Stability of a processor-sharing queue with varying throughput. *J. Appl. Prob.* **45**, 953–962.
[35] MOYAL, P. (2013). On queues with impatience: stability, and the optimality of earliest deadline first. *Queueing Systems* **75**, 211–242.
[36] MOYAL, P. (2015). A generalized backward scheme for solving non-monotonic stochastic recursions. *Ann. Appl. Prob.* **25**, 582–599.
[37] MOYAL, P., Bušić, A. and Mairesse, J. (2021). A product form for the general stochastic matching model. *J. Appl. Prob.* **58**, 449–468.
[38] MOYAL, P. AND PERRY, O. (2017). On the instability of matching queues. *Ann. Appl. Prob.* **27**, 3385–3434.
[39] MURDOCH, D. AND GREEN, P. (1998). Exact sampling from a continuous state space. *Scand. J. Statist.* **25**, 483–502.
[40] NAZARI, M. AND STOLYAR, A. L. (2019). Reward maximization in general dynamic matching systems. *Queueing Systems* **91**, 143–170.
[41] PROPP, J. G. AND WILSON, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures Algorithms* **9**, 223–252.
[42] RAHME, Y. AND MOYAL, P. (2021). A stochastic matching model on hypergraphs. *Adv. Appl. Prob.* **53**, 951–980.
[43] THORISSON, H. (2000). *Coupling, Stationarity, and Regeneration*. Springer, New York.
[44] WILSON, D. B. (2000). How to couple from the past using a read-once source of randomness. *Random Structures Algorithms* **16**, 85–113.